Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/CA05/000194

International filing date: 16 February 2005 (16.02.2005)

Document type: Certified copy of priority document

Document details: Country/Office: CA
Number: 2,457,909
Filing date: 16 February 2004 (16.02.2004)

Date of receipt at the International Bureau: 06 April 2005 (06.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

Office de la propriété intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An Agency of Industry Canada

PCT/CA 2005/000194
15 MARCH 2005 15·03.05

Bureau canadien des brevets

Certification

La présente atteste que les documents ci-joints, dont la liste figure ci-dessous, sont des copies authentiques des documents déposés au Bureau des brevets.

Canadian Patent Office

Certification

This is to certify that the documents attached hereto and identified below are true copies of the documents on file in the Patent Office.

Specification and Drawings, as originally filed, with Application for Patent Serial No: **2,457,909**, on February 16, 2004, by **CHRISTOPHER DAVIES,** for "Method and System for Self-Organizing Reliable, Multiple Path Data Flow Transmission of Data on a Network".

Agent certificateur/Certifying Officer

March 15, 2005
Date

Canada

(CIPO 68)
31-03-04

OPIC CIPO

ABSTRACT OF THE DISCLOSURE

A system and method for self-organizing, reliable, multiple path data flow transmission of messages data on a network uses queues to transmit messages between end-user modules (EUMs) on nodes on the network. The EUMs include the end user applications with which queues are associated. A network communications manager (NCM) resident on every node manages all transmission of messages between nodes.

The NCM on a given node only has knowledge of nodes that are neighbor nodes to that given node, but has knowledge of all queues associated with all EUMs. Messages are divided into EUM messages, which are placed in queues by the NCM on each node, and system messages, which are not placed in queues but are used by the NCM to determine when and where, i.e. to which neighbor node, messages may be sent. The NCM on each node chooses a neighbor node as a target node for sending EUM messages for each queue, based on the best node latency and at capacity status of each neighbor node. These target nodes are used to provide potential routes to queues and multiple path data flow for queues that carry EUM data messages for user applications. These target nodes are constantly updated to provide the best paths on an adaptive basis and to ensure that all paths are valid, improving network reliability.

When choosing when to send data to a target node, each node uses tokens for flow control to ensure that target nodes do not become overloaded. The node also compares node latencies for multiple target nodes to ensure that the lowest node latency target node is chosen.

By using neighbor nodes as target nodes, node latency, and at capacity information for determining when and where to send data, there is no need to maintain any global knowledge of all paths in the network. Further, the constant updating of target nodes ensures that the network maintains optimal and valid paths for messages, thus ensuring efficiency and reliability. Finally,

the constant updating of target nodes ensures that reliability and efficiency are provided on an adaptive, self-organizing basis.

CA 02457909 2004-02-16

B&P File No.11199-7

BERESKIN & PARR **CANADA**

Title: **METHOD AND SYSTEM FOR SELF-ORGANIZING RELIABLE, MULTIPLE PATH DATA FLOW TRANSMISSION OF DATA ON A NETWORK**

Inventor(s): **Christopher Davies**

SYSTEM AND METHOD FOR SELF-ORGANIZING, RELIABLE, MULTIPLE PATH DATA FLOW TRANSMISSION OF DATA ON A NETWORK

FIELD OF THE INVENTION

[0001] The present invention relates to transmission of messages on a network.

BACKGROUND OF THE INVENTION

[0002] In a data network, data from an initiating node is transmitted from node to node throughout the network until it reaches a terminating node. In a conventional data network, this process often requires establishment and maintenance of an overall map or network topology of all of the nodes in the network to calculate data paths from the source node to the destination node. At the same time, to maximize efficiency, it is desirable to choose the best data path or paths for sending data to ensure maximum bandwidth use and the shortest delivery time from source node to destination node. Since ongoing data transmission throughout the network will affect the capacity of data transmission and reception for each node, available data paths, as well as the best choice of data paths, may be subject to constant change. Thus, in a conventional network where an overall network topology is maintained to calculate data paths, in addition to maintaining a map of all available nodes and data paths, it may also be necessary to maintain repositories of information on the capacity status of all nodes to ensure selection of the best data path or paths. Such a system involves significant processing overhead to maintain the map of nodes and routes as well as monitor node status. In addition, maintenance of such information in one or more centralized locations may involve significant amounts of bandwidth use in order to constantly update the node and path status.

The present invention addresses all of these issues. It provides minimum latency paths for transmission of messages without requiring any maintenance of global network topology. It is therefore self-organizing. Further, the invention constantly optimizes and verifies validity of paths for

transmission of messages. Thus, the present invention further provides adaptive and reliable transmission of messages.

SUMMARY OF THE INVENTION

[0003] A system for transmission of messages between nodes on a network, said system comprising:

(a) a plurality of queues on each node; and

(b) a network communication manager on each node, wherein said network communication manage has knowledge of neighbour nodes and knowledge of all queues on each node.

[0004] A method for determining the best path through a network comprising the steps of:

(a) determining the latency and capacity of neighbour nodes and selecting the most efficient neighbour node to receive a message; and

(b) repeating step (a) on a regular basis.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] For a better understanding of the present invention, and to show more clearly how it may be carried into effect, reference will now be made, by way of example, to the accompanying drawings which aid in understanding an embodiment of the present invention and in which:

[0006] Figure 1 is a block diagram of a network in which the present invention may be implemented;

[0007] Figure 2 is a block diagram of the modules comprising a node;

[0008] Figure 3 is a block diagram of an end-to-end connection;

[0009] Figure 4 is a flowchart of two steps of a method embodying the present invention;

[0010] Figure 5 is a flowchart of the steps for determining where to send messages.

[0011] Figure 6 is a flowchart of the steps for processing latency updates.

[0012] Figure 7 is a flowchart of the steps for processing a new queue.

[0013] Figure 8 is a flowchart of the steps for updating the target nodes for a known queue when receiving a latency update.

[0014] Figure 9 is a flow chart of the steps for propagating a new queue and establishing potential routes for the new queue.

[0015] Figure 10 is a flowchart of the steps for establishing an end-to-end connection and data flow.

[0016] Figure 11 is a flowchart of the steps for maintaining and updating data flow and potential routes.

[0017] Figure 12 is a flowchart of the steps for maintaining at least one target node for each queue on each node.

[0018] Figure 13 a flowchart for the steps of the loop globally unique identifier test.

[0019] Figure 14 is a flowchart of the steps for optimizing the data flow in an end-to-end connection.

[0020] Figure 15 is a flowchart of the steps removing a target node for a data flow node.

[0021] Figure 16 is a flowchart of the steps for removing a node completely from all paths on the data flow for a terminating queue.

[0022] Figure 17 is a flowchart of the steps for optimizing potential routes to a queue of any type for a node outside of the data flow for an end-to-end connection.

[0023] Figure 18 is a flow chart of the steps for resolving loops created in nodes outside the data flow.

[0024] Figure 19 is a flowchart of the steps for determining when a node may send messages to another node.

[0025] Figure 20 is a flowchart of the steps for flow control.

[0026] Figure 21 is a flowchart of the steps for determining which data flow nodes are useful target nodes using a latency comparison.

DETAILED DESCRIPTION OF THE INVENTION

[0027] The present invention relates to transmitting messages on a network. The network is comprised of a series of nodes. Each node uses queues as the vehicle for transmitting messages containing data to neighbor nodes. Queues are associated with end-user modules (EUM), which contain end-user applications.

[0028] For each queue, a node uses latency information to designate one or more neighbor nodes as target nodes for transmitting messages for a particular queue. Target nodes for a given queue are the only nodes to which a node will transmit messages on that queue for the associated EUM. The target nodes, when chained together allow for end-to-end connections (EEC) from an initiating node to a queue on the node, referred to as the terminating node, that houses the EUM for which the queue was created. All nodes use data structures known as tokens to manage flow control between each other which, along with latency comparisons, assist in determining when a node may send a message to another node.

[0029] To assist the reader in better understanding the terminology used in the description of the present invention, definitions of some of the terms and abbreviations used in the description are provided in Table 1.

Table 1

Term	Definition
Acknowledgement messages	Messages sent from a terminating node of an EEC to the initiating queue on that EEC's initiating node to indicate that EUM data messages sent to the terminating queue have been received on the terminating node. Acknowledgement messages are EUM messages and travel over the potential route between the initiating node and terminating node.

Term	Definition
CTNTP	Current target node time period: the period of time for which the node latency of the current chosen destination is examined for a latency loop test.
Data flow node	A node that is part of the path of target nodes for a terminating queue for an EEC leading from the initiating node to the terminating node for that EEC. Any target node of a data flow node is also a data flow node for that terminating queue.
EEC	End-to-end connection: a path of target nodes for transmitting EUM data messages and EEC messages from an initiating node to a terminating queue on the terminating node and a path of target nodes for sending acknowledgement messages and EEC messages to the initiating queue associated with the EUM that requested the EEC connection on the initiating node. EECs are requested by sending an EEC message requesting the EEC to the information queue. The path for the terminating queue created on the terminating node is the data flow for that EEC. The path for the initiating queue is a potential route.
EEC message	End-to-end connection message: a message containing information for establishing and maintaining an EEC. EEC messages are EUM messages and are transmitted via initiating queues, information queues, and terminating queues.
EUM	End-user module: a module that houses an end-user application.
EUM data message	End-user module data message: a message containing data for use by an end-user application contained in an EUM. EUM data messages travel on terminating queues and are EUM messages.
EUM message	End-user-module message: A message which travels through queues on the network and which contains data for use by end-user modules or relating to establishment and maintenance of end-to-end connections. EUM messages include EUM data messages, EEC messages, and acknowledgement messages.
GUID	Globally unique identifier: an identifier used to uniquely identify queues and loop test messages.
HSPP	High speed propagation path: a priority path for reaching the core of the network.

Term	Definition
Information queue	A queue for informing other nodes that an EUM is ready to receive messages for a given purpose over an EEC. Information queues may carry EEC messages and are used only for establishing EECs.
Initiating node	The node that houses the EUM for which an initiating queue is created. The initiating node, which is an origin node, is the node where requests for EECs are initiated.
Initiating queue	Connection Acknowledgment Queue: a queue for receiving acknowledgement messages and EEC messages for EUM data messages sent over an EEC to a terminating queue associated with an EUM on a terminating node. The initiating queue is initially created on an initiating node that houses the EUM , associated with the initiating queue that requested the EEC.
MNQA	Maximum new quota allotment: the maximum amount of new tokens that a receiving node will allot to a sending node at any time.
NCM	Network communications module: a module resident on a node that manages transmission of messages to other nodes.
Neighbor node	A node to which another node may send a message by traversing only one data link.
OCTC	Over capacity token count: the number of bytes over what is required to keep a node at capacity when that node is already at capacity.
Origin node	The node that houses an EUM for which a queue is created. There are two types of origin node: terminating nodes for information queues and terminating queues and initiating nodes for initiating queues.
OTE	Outstanding tokens estimate: an estimate of the number of tokens that a sending node currently possesses, as calculated by the receiving node.
OTL	Outstanding token limit: the maximum amount of tokens that a receiving node will allow a sending node to possess at any given moment.
PTNTP	Potential target node time period: the period of time for which the node latency of the current chosen destination is examined for a latency loop test.
Target node	A directly connected node, associated with a particular queue, to which a given node may send EUM messages for

Term	Definition
	that queue.
Terminating node	The node housing the EUM for which a terminating queue and information queue are created. A terminating node is an origin node.
Terminating queue	A queue for receiving EUM data messages and EEC messages sent by an EUM on an initiating node over an EEC for messages sent over an EEC to a terminating queue associated with an end user module. The terminating queue is initially created on the terminating node, which houses the EUM, associated with the terminating queue, for which the EUM data messages and EEC messages are targeted.

[0030] Reference is first made to Figure 1, a block diagram of a network, shown generally as 5, in which the present invention may be implemented. Figure 1 will be used as a basis for explaining the terminology of the present invention and for providing examples.

[0031] The network 5 comprises a plurality of nodes 10. The nodes 10 may be computers, switches, microprocessors, cellular phones, personal digital assistants or the like. Each node 10 in the network 5 is connected to one or more nodes 10 by the data links 15. The data links 15 within the network 5 may be comprised of wireline connections, wireless connections, data bus connections, or the like. Messages are physically sent over the network 5 via the data links 15 between the nodes 10.

[0032] The network 5 shown is exemplary in that there may be any number of nodes 10 in the network 5. The types of devices listed as the nodes 10 and the various sorts of data links 15 mentioned are enumerated for purposes of example. It is not the intention of the inventor to limit the nodes to a specific type or platform or the data links to a given type or protocol.

[0033] A node 10 may have different status depending on it's relationship to other nodes 10 and the messages that the node 10 is currently processing or transmitting. This status may constantly change. However, the

various modules that may be available in each node 10 in the present invention are the same. Thus, in this description, reference is made to node 10 in all cases, regardless of status. Additional terms will be used to indicate the particular status of a node 10 at a given moment.

[0034] A node 10 will only transmit a message of any type to a neighbor node 10. A first node is a neighbor node 10 for a second node 10 if it can be reached by the second node 10 through one data link 15. Each node 10 is aware only of it's neighbor nodes 10.

[0035] To better illustrate the concept of neighbor nodes, reference is made to Figure 1 in conjunction with Table 1. For example, nodes 10a and 10b are neighbor nodes 10 for each other since they can reach each other by traversing only one data link 15a. Nodes 10b and 10c are also neighbor nodes 10, as they may reach each other by traversing only data link 15b. However, nodes 10a and 10c are not neighbor nodes because, for 10a to reach 10c or for 10c to reach 10a, data links 15a and 15b, at the very least, must be traversed. Table 1 shows the neighbor nodes 10 for each node 10 in Figure 1.

Table 1

Node	Neighbor nodes
10a	10b
10b	10c, 10g
10c	10b,10d,10f
10d	10c,10e
10e	10d,10f
10f	10c,10e,10g
10g	10b,10f

[0036] Reference is now made to Figure 2, a block diagram of the modules comprising a node 10. Each node 10 contains a network communications module (NCM) 20, which is responsible for all transmission and reception of all messages for the node 10 to and from other nodes 10 in the network 5. The NCM 20 physically transmits and receives all messages to and from other nodes 10 via a network transmission means (NTM) 25, which provides access to the data links 15 for physically sending messages. A NTM 25 may be a network card, antenna for mobile computing or cellular phone, modem, or any other means capable of interfacing with data links for physically transmitting data. There may be multiple NTMs 25 on each node 10. The types of devices mentioned as NTMs 25 are examples only. It is not the intention of the inventor to limit what may constitute NTMs 25 on a node 10 to any specific device.

[0037] Each node 10 may optionally contain one or more EUMs 30 for user applications. Each node 10 may also contain one or more information queues 35, terminating queues 40, and initiating queues 45, which are used as destinations for transmission of messages for one EUM 30 to another EUM 30. For purposes of brevity, the term queue will refer to information queues 435, terminating queues 40, and initiating queues 45 in all circumstances where they receive identical treatment. The specific terms information queue 35, terminating queue 40, or initiating queue 45 are used where only an information queue 35, terminating queue 40, or initiating queue 45 is the subject of the reference. All queues are associated with an EUM 30.

[0038] While the NCM 20 on each node 10 is only aware of the neighbor nodes 10, the NCM 20 is aware of all queues and maintains a version of each queue on it's node 10. Thus, while every queue is associated with an EUM 30, it is not necessary that an EUM 30 and the associated queues be present on the same node 10. The NCM 20 on each node 10 also creates and maintains a set of data structures 50 for each queue on the node 10. These data structures 50 are used, among other things, for recording

node latency and at capacity status and choosing which neighbor nodes 10 will be sent messages.

[0039] The designation of a queue as an information queue 35, terminating queue 40, or initiating queue 45 depends on it's relationship to the EUM 30 for which it was created. An information queue 45 is used solely to provide information to other EUMs 30 on other nodes 10 that a terminating EUM 30 is ready to receive messages for a given purpose. Other EUMs 30 then use the information queue 35 to establish end-to-end connections (EECs) to the terminating EUM 30. Once an EEC is established, a terminating queue 40, associated with the terminating EUM 30, is established for transmission of messages to the terminating EUM 30 by an initiating EUM 30. The initiating EUM 30 is the EUM 30 that requests the EEC and for which the EEC is specifically established. An initiating queue 45, associated with the initiating EUM 30 is also created for receiving messages sent by the terminating EUM 30 to inform the initiating EUM 30 that messages sent by the initiating EUM 30 have been received. Thus, when a terminating EUM 30 is ready to receive messages from another EUM 30 on a different node 10 for a given purpose, there is one information queue 35 created for the terminating EUM 30 for that purpose. Then, initiating EUMs 30 can send messages to the information queue 35 to establish EECs to the terminating EUM 30 for the purpose associated with the information queue 35. Each EEC established will result in the creation of a terminating queue 40 and initiating queue 45 for that particular EEC.

[0040] The status of a node 10 changes based it's relationship to other nodes 10 and the messages being transmitted. More specifically, a node's 10 status is based on it's relationship to queues that carry messages for EUMs 30. A node 10 that houses an EUM 30 for which a queue was created is an origin node 10 for that queue. With regard to an EEC, the node 10 that houses the terminating EUM 30 that is the intended recipient of the messages sent through the terminating queue 40 of an EEC is a terminating node 10 for that particular EEC and the origin node 10 for that EECs terminating queue

40. The terminating node 10 for an EEC is also the origin node 10 for an information queue 35, which is used for establishing EECs to the terminating EUM 30. A node 10 that houses the initiating EUM 30, for which an EEC is created for sending messages to a terminating EUM 30, is referred to as an initiating node 10 for that EEC. The initiating node 10 for an EEC is also the origin node 10 for that EEC's initiating queue 45 used for messages sent from the terminating EUM 30 to the initiating EUM 30.

[0041] As mentioned, the NCM 20 on each node 10 only maintains knowledge of the neighbor nodes 10, but maintains knowledge of all queues in the network 5. To maintain knowledge of all queues, the NCM 20 maintains a corresponding queue on it's node 10 for every queue of which it is informed. Further, for each queue, the NCM 20 maintains a list of neighbor nodes 10, referred to as target nodes 10, to which it will send messages, on behalf of an EUM 30, to be transported in that queue. Only the NCM 20 on the originating node 10 for a queue knows that it's node 10 is the originating node 10 for that queue. All other nodes 10 consider their target nodes to be the originating node 10 for the queue.

[0042] When an initiating EUM on an initiating node 10 wishes to send EUM messages to a terminating EUM 30 on a terminating node 10, it sends the messages to it's target nodes 10 for that queue. These target nodes 10 in turn send the message to their target nodes 10 for that queue. This process continues, with the message being sent from target node 10 to target node 10 for the queue until the origin node 10, either a terminating node 10 or initiating node 10, for the queue is reached. The path of target nodes from an initiating node 10 to a terminating node 10 associated with a terminating queue 40 for an EEC is referred to as the data flow for that terminating queue 40. Target nodes 10 in the data flow have the status of data flow nodes 10 for that terminating queue 40. The path of target nodes from any node 10 to an origin node 10 for an initiating queue 45 or information queue 45 is referred to as a potential route for that queue. The path of target node 10 for a terminating

queue from any node 10 outside the data flow, i.e. a node 10 that is not a data flow node 10 for that terminating queue 40, is also a potential route.

[0043] Whereas each node 10 on a potential route for a queue may only have one target node 10, data flow nodes 10 for a terminating queue 40 for an EEC may have multiple target nodes 10. Thus, the data flow for a terminating queue 40 may comprise multiple paths to the terminating node 10 for that EEC. The potential route to the terminating node 10 for a terminating queue 40, if that node 10 is not in the data flow, as well as all nodes 10 to the origin node 10 for an information queue 35 or initiating queue 45 is comprised of only one path.

[0044] There are two general categories of messages utilized by the present invention, EUM messages and system messages. EUM messages include EUM data messages, EEC messages, and acknowledgement messages. EEC messages are used to establish and maintain end–to-end connections between EUMs 30 on initiating nodes 10 and terminating nodes 10. EEC messages are carried on all three types of queues and may be transported on target nodes 10 in a potential route or in the data flow for the terminating queue 40 between the initiating node 10 and the terminating node 10 for an EEC. EUM data messages, which contain user application data sent from an initiating EUM 30 to a terminating EUM 30 over an EEC, are transmitted in terminating queues 35. Thus, they are only transported over data flow nodes for the terminating queue 40 for a particular EEC. Acknowledgement messages, used to acknowledge receipt by a terminating EUM 30 of EUM data messages sent by the initiating EUM 30 for an EEC, are transmitted in initiating queues 45 on target nodes 10 forming the potential route between the terminating node 10 and the initiating node 10 for that EEC.

[0045] System messages are used to transmit information about node latency, at capacity status, tokens, and neighbor nodes 10 to which a node 10 may pick as target nodes 10 to send EUM messages. While system messages may relate to queues, they are not actually placed in queues. Rather, they are processed and generated by the NCM 20 to manage

transmission of EUM messages sent through queues. Thus, they may be transmitted from the NCM 20 on a first node to the NCM 20 on a second node 10 that is a neighbor node 10 to the first node 10, even if the second node 10 is not a target node 10 of the first node 10. In fact, system messages are used to determine target nodes 10 and thus where a node 10 may send EUM messages. System messages are also used for flow control which determines when a node 10 may send messages of any kind to another node 10, including EUM messages for a given queue on a given node 10.

[0046] The NCM 20 performs all calculations for system messages, generates the system messages, and sends the system messages, via the NTM 25, to neighbor nodes 10. The NCM 20 also generates and sends EEC messages via the NTM 25, at the request or instruction of an EUM 30. The NCM 20 is further responsible for grouping messages of all types before sending, via the NTM 25, to other nodes 10. When a node 10 receives a message, it is first inspected by the NCM 20, which reads the message from the NTM. The NCM 20 then separates EUM messages from system messages, forwards the EUM messages on to the appropriate queue if required, and processes the system messages received from the neighbor node 10. The NCM 20 may break EUM messages destined for a queue into smaller messages for purposes of storage in the queue or for sending the smaller messages to the same queue, via the NTM, on multiple neighbor nodes 10. The NCM 20 may also combine EUM messages for a given queue received from multiple neighbor nodes 10 into larger EUM messages to ensure that the queue is carrying the largest EUM message possible for that queue.

[0047] The NCM 20 performs all operations necessary for managing the transmission and reception of messages via the queues. As such, the NCM 20 creates and maintains all queues required by EUMs 30 and manages all aspects of the use of queues for transporting messages between nodes 10, including latency updates, flow control, and establishment and maintenance of potential routes and data flow. Thus, when an EUM 30

wishes to transmit or receive messages, it sends a request to the NCM 20. The NCM 20 then creates the required queues and the data structures 50 for each queue. Each queue has it's own set of data structures 50 associated with it.

[0048] When a first EUM 30 sends messages to a second EUM 30, the first EUM 30 initially sends the messages to the NCM 20 which routes the message to the appropriate queue associated with the second EUM 30 and handles transmission of the message for that queue, via the NTM 25, to target nodes 10. The NCM 20 on each subsequent target node 10 transmits the messages to it's target nodes until the messages reach the origin node 10, whether terminating or initiating node 10 for the queue where the second EUM 30 is resident.

[0049] When an EUM message for a queue is received by a node 10, the NCM 20 governs the handling of the EUM message for the queue. When a first node 10 receives an EUM message sent by an EUM 30 on a second node 10, the NCM 20 on the first node 10, after reading the message from the NTM 25, processes the EUM message and places it in the queue associated with the EUM 30 for which the EUM message is destined. If the first node 10 is the origin node 10 for the queue, the EUM 30 can then access the EUM message in the queue.

[0050] The data structures 50 maintained by the NCM 20 on a node 10 for each queue include a list of target nodes 10 and a list of all neighbor nodes 10 and the queues 35 for which each neighbor node 10 is a target node 10. The data structures 50 further comprise variables to record the latency of each neighbor node 10 and the node's 10 own node latency, variables to record the at capacity status of each neighbor node 10 and the node's 10 own at capacity status. In addition, the NCM 20 on each node maintains variables to variables to track tokens for flow control. Further, for each EEC, the NCM 20 on each node 10 maintains flag variables for the terminating queue 40, to indicate whether it's node 10 is in the data flow. Also, the NCM 20 on the initiating node 10 for an EEC maintains a buffer for

the initiating queue 45 and the NCM 20 on the terminating node 10 for an EEC maintains a buffer for the terminating queue 40.

[0051] To assist the reader in better understanding the different possible statuses of nodes 10, reference is now made to Figure 3, a block diagram of an EEC. The EEC and potential routes and data flow is demonstrated by using four nodes 10h, 10i, 10j, 10k. Suppose that node 10j houses an EUM 30 and has created an information queue 35 associated with that EUM 30. Suppose further that an EEC has been established using the information queue on 10j between an initiating EUM 30 on 10h to send EUM messages to 10j. Thus, the EUM 30 on 10j is the terminating EUM 30 for the EEC and the EUM 30 on 10h is the initiating EUM 30. 10j is the origin node 10 for the terminating queue 40 and information queue 35 and the terminating node 10 for the EEC. 10h is the origin node 10 for the initiating queue 45 and the initiating node 10 for the EEC.

[0052] Suppose that the NCM 20 on each node 10 has chosen the target nodes 10 for the information queue 35 as shown in Table 3. The potential routes for each node 10 to the origin node 10i for the information queue 35 are as shown in Table 3 and as a dotted and dashed line in Figure 3. For example, since node 10k has chosen node 10h as a target node 10 for the information queue 35, 10h has chosen 10i as a target node 10, and 10i has chosen 10j as a target node 10, the potential route from 10k to 10j for the information queue 35 is 10h, 10i, 10j.

Table 3

Node	Target node: information queue	Potential route: Information Queue (Left to right order).
10h	10i	10i, 10j
10i	10j	10j
10j	None	None
10k	10h	10h,10i,10j

[0053] Table 4 is used to demonstrate the data flow and potential routes to the terminating node 10 for the terminating queue 40 for the EEC. The data flow is shown as a bold dashed line in Figure 3. Suppose, as indicated in Table 4, that 10i has been chosen as a target node 10 for node 10j for the terminating queue 40 and that 10h, the initiating node 10 for the EEC has chosen 10i. Since 10i is the initiating node 10 for the EEC, all nodes that are part of the path of target nodes 10 from 10i to 10j for the terminating queue 40 for the EEC are in the data flow. Thus the entire data flow for the terminating queue 40 on the EEC is 10h,10i, 10j. 10i and 10j are therefore data flow nodes 10 and any nodes 10 that these nodes 10i, 10j subsequently select as target nodes 10 for the terminating queue 40 would also become data flow nodes 10 for that terminating queue 40. Thus, were 10k to have been chosen as a target node 10 of 10h for the terminating queue 40, node 10k would also be in the data flow. The selection of 10k would therefore provide and an additional path in the data flow commencing at the initiating node 10h, passing through 10k, and then finishing at the terminating node 10j. However, since 10k has not been chosen as a target node 10 for 10h or any other data flow node 10, 10k merely has a potential route for the terminating queue 40 to 10j. This potential route, namely 10k to 10j, is shown as a bold dashed line in Figure 3.

Table 4

Node	Target node: terminating queue	Path of target nodes: Terminating queue (Left to right order).	Type of path: potential route or data flow to terminating node for terminating queue (left to right order):
10h	10i	10i, 10j	Data flow
10i	10j	10j	Data flow
10j	None	None	None
10k	10j	10j	Potential route

[0054] Table 5 shows the target nodes 10 for the initiating queue 45 and the resulting potential routes to the initiating node 10h for the initiating queue 45 for each node 10. The potential routes from each node 20 are shown as dotted lines in Figure 3.

Table 5

Node	Target node: Initiating queue	Potential routes: Initiating node for initiating queue (left to right order).
10h	None	None
10i	10h	10h
10j	10k	10k, 10h
10k	10h	10h

[0055] Having explained the various types of queues used, potential routes, EECs, and the different statuses that may be accorded a node 10, a brief explanation of how nodes 10 maintain knowledge of neighbor nodes 10 and queues is provided. These tasks are assumed by the NCM 20 on each node 10.

[0056] The NCM 20 on a node 10 maintains a record of all neighbor nodes 10. This may be done by tracking the data links 15 to and from the given node 10 or by variables designating each node 10 that is a neighbor node 10. Any time a new node 10 is added to the network 5, it's NCM 20 broadcasts it's presence, via the NTM 25, over it's data links 15. thus informing all neighbor nodes 10 of the new node's 10 existence. The neighbor nodes 10 for the new node 10 do likewise thus informing the new node 10 of their presence. The methods proposed for tracking neighbor nodes are suggestions only. Other methods are possible. It is not the

intention of the inventor to restrict the tracking of neighbor nodes 10 to a particular method.

[0057] As for knowledge of queues, each queue when first created by the NCM 20 on it's origin node is assigned a queue label that includes a reference to the EUM 30 for which the queue was created and a globally unique identifier (GUID). For example, such a unique label could be of the format:

```
Struct queue label
{
integer EUMQueueNumber
String  GUID
}
```

EUMQueueNumber is an integer, such as 123456, and the GUID is a unique string, such as af9491de5271abde526371. This would yield the following queue label: 123456.af9491de5271abde526371. A label could also have several integers used to identify the particular queue using the following format:

```
Struct queue label
{
integer EUMAppID.EUMQueueNumber.GUID
integer EUMQueueNumber
string  GUID
}
```

In this example, the queue label contains an additional field EUMAppID that could specifically identify the associated EUM 30.

[0058] As one skilled in the art will recognize, alternative implementations for the queue label are also possible, provided that the queue label is unique in that it only designates one queue associated with one

EUM 30 for every node in the network 5. It is not the intention of the inventors to restrict the labels used for queues to a given format.

[0059] If the NCM 20 on a node 10 has knowledge of a queue, that node's 10 NCM 20 will transmit a message about that queue, including the queue label, to all neighbor nodes 10 of that node 10. The NCM 20 on each neighbor node 10 will in turn create a queue having the same queue label associated with that queue. The NCM 20 on each neighbor node 10 then informs all neighbor nodes 10 of the neighbor node 10 of the queue. This process continues until every node 10 is made aware of any queue that has been created and has a corresponding queue with an identical queue label. As for origin nodes 10, there are a variety of methods by which an NCM 20 on an origin node 10 can recognize that an EUM message is destined for an EUM 30 resident on that node. For example, the NCM 20 could verify the part of the queue label that identifies the EUM 30. Alternatively, the NCM 20 on the origin node 10 could designate the origin node 10 itself as the target node 10 for the queue. However, it is not the intention of the inventors to restrict the scope of the invention to any particular manner in which an origin node 10 will recognize itself as such for a given queue.

[0060] It should be noted that, as queues serve as the software vehicle for transporting EUM messages in the network 10, they play a similar role to that of ports in a traditional Internet protocol (IP) network. However, unlike a traditional IP network, to send an EUM message to an EUM 30 on an origin node 10, the NCM 10 on the node 10 sending the EUM message only needs to know the label of the queue, and not an IP address and port number for the EUM 30 on the origin node. At the same time, in the present invention, the NCM 20 on a node 10 does not need to know about any other nodes 10 except neighbor nodes 10, as opposed to a conventional IP network where knowledge of all nodes 10 in the network must be maintained. Thus, the present invention, by using queues, avoids having to maintain any detailed topological information about the network 5 while still providing full network

functionality for end-to-end transmission and reception of EUM messages. A node 10 only knows of queues and it's neighbor nodes 10

[0061] Figure 4 is a flowchart of two steps of a method embodying the present invention. Shown at 55, one step of the method consists of determining where a node 10, among it's neighbor nodes 10, may send EUM messages. This is governed by limiting node knowledge to neighbor nodes 10 and the designation of target nodes 10 and data flow nodes 10. Referring to the other step 60, the method also determines when each node 10 may send messages to a neighbor node 10. This is effected by flow control and by choosing which target nodes 10 are most appropriate for sending a message at a given time using a latency comparison. Both procedures influence each other in that procedure 55 determines which neighbor nodes 10 are target nodes 10 for sending EUM messages to queues at any given time. Procedure 60 is then used for deciding when it is appropriate to use a given target node 10, as a data flow node 10, to send an EUM message to a terminating queue 40, as well as, more generally, when a node 10 can send any message to a neighbor node 10. If a given target node 10 cannot be used at a given time, then procedure 55 may again have to be invoked to examine which neighbor nodes 10 may be available as new target nodes 10.

[0062] Since each node 10 will only send EUM messages to nodes 10 designated as target nodes 10 for a queue, it is the information used for choosing the target nodes 10 that determines the path of messages, whether on a potential route for the queue or in the data flow for terminating queues 40. Thus, the information used for designating target nodes 10 determines to which neighbor nodes 10 EUM messages for a queue can be sent.

[0063] The NCM 20 on each node 10 determines the target nodes 10 for a queue based on node latency and "at capacity" status. This information is constantly updated by the NCM 20 on a node 10 based on latency updates, which contain both node latency and at capacity status, received from neighbor nodes 10. In addition, token updates, which are used for flow control to ensure a node 10 or a queue on a node 10 is not overrun with

messages, also contain latency updates. Tokens are stored in variables, which represent the number of tokens as integers. In other words, a token value of 100 means 100 tokens. A node 10 that wishes to transmit messages of any kind to a neighbor node 10 must always have tokens for the neighbor node 10 and, for EUM messages, for the specific desired queue on the neighbor node 10. Thus, the token updates relate to the timing of when a node 10 may send messages to a neighbor node 10. In addition, for data flow nodes 10 for an EEC, latency comparisons are used to determine when a given data flow node 10 may use a given target node 10 for sending EUM data messages or EEC messages.

[0064] Latency updates are sent through the network 5 as a constant stream in a throttled manner so as not to exceed a given percentage of node-to-node bandwidth. Typically, the percentage would be 1 to 3 percent, although other values are possible. The NCM 20 on each node 10 cycles through all known available node latencies and at capacity status information for a queue in a round robin fashion. However, the NCM 20 on a node 10 may pre-empt the processing and sending of latency updates for a first queue that the node 10 would normally perform as part of round robin queue processing in order to immediately send a latency update for a second queue. When sending a latency update, the NCM 20 calculates the latency update and sends the latency update for the queue, or latency updates for groups of queues, to each neighbor node 10. Other ways to determine what order or frequency to send latency updates could include: percentage change in node latency, marking a particular class of queue labels for more frequent updates, or a distance from data flow counter could be used to increase latency updates to nodes 10 in the vicinity of data flow nodes 10. Other solutions are also possible, it is not the intention of the inventors to restrict the method for determining the timing of latency and capacity updates.

[0065] Reference is now made to Figure 5, a flowchart of the steps for determining where to send messages. As such, Figure 4 is an expanded view of step 55 of Figure 4. Commencing at step 65, each node 10 receives

latency updates. Turning now to step 70, the information in the latency updates is used to propagate knowledge of queues throughout the network 5. At each node 10, a neighbor node 10 is chosen as a target node 10 for the new queue, thus establishing potential routes. Step 70 occurs for all new queues of all types. Next, at step 75, if an information queue 35 has been propagated, it is possible for EUMs 30 on initiating nodes 10 to request an EEC connection, via their NCMs 20, to the EUM 30 for which the new information queue 35 was created. This will result in the creation and propagation of a terminating queue 40 and initiating queue 45. Using the terminating queue 40, the EEC and the data flow for the EEC's terminating queue 40 will be established. A potential route to the initiating node 10 for the initiating queue 45 will also be established for EEC messages and acknowledgement messages sent from the terminating node 10. The propagation of the initiating queue 45 and terminating queue 40 and choice of target nodes 10 for them also requires latency updates, as does the marking of target nodes 10 for the terminating queue 40 as data flow nodes 10. Thus, the information in latency updates also is essential for establishing EECs and data flow. Finally, at step 80, once potential routes and data flow for EECs are established, latency updates are used to update and maintain them.

[0066] When using latency updates for choosing a target node 10 for a queue, the NCM 20 on a node 10 will attempt to choose a target node 10 that respects the following criteria:

(a) The target node 10 should provide the lowest latency for the queue to the origin node 10;

(b) The target node 10 should not be at capacity; and

(c) The target node 10 should not create a loop.

[0067] For the purposes of the present invention, a loop is defined as a path of target nodes 10 wherein, if an EUM message for a queue is sent from a node 10 on the path of target nodes10, whether data flow or potential route, the EUM message will finish at the same node 10 from where it was originally sent. Thus, the EUM message will not reach the origin node 10 for the queue.

An instant loop is a loop consisting of two neighbor nodes 10, which have been designated as target nodes 10 for each other for the same queue by their respective NCMs 20.

[0068] For the purposes of the present invention, node latency for a queue refers to the amount of time that it would take for a message for that queue sent by the node 10 that sends the latency update to arrive at the node 10 that is the origin node 10 for the queue and to be de-queued by the EUM 30 on the origin node 10. Node latency, represented by the variable NLatency, for a queue is calculated by the NCM 20 on the node 10 sending the latency update by using the following node latency formula:

NLatency = MinOverTimePeriod (BytesInQueue * BytesPerSecondSendRate + ServiceTimeOnQueue + MIN (NLatencyTargetNode + PhysicalNetworkLatencyTargetNode)).

[0069] MinOverTimePeriod is a period of time determined by the time it takes to perform a minimum of, for example, five sends or receives from the node 10 broadcasting the latency update and the node 10 receiving the latency update for the queue associated with the latency update. It is also a minimum time of, for example, 30ms (or a reasonable multiple of the granularity of the operating system timer). BytesPerSecondSendRate is the best estimate of the rate of data flowing out of the queue on the node 10 broadcasting the latency update. NLatencyTargetNode 10 is the node latency recorded by the node 10 broadcasting a latency update for an individual target node 10 for a queue. The exact minimum number of sends or receives or minimum time period for MinOverTimePeriod may vary depending on requirements of the network 5, EUMs 30, or NCM 20 on each node10. It is not the intention of the inventors to limit the minimum number of sends and receives or time period to the examples provided.

[0070] PhysicalNetworkLatencyTargetNode is the time needed to send a packet similar to the average message size from the node 10 broadcasting the latency update to a target node 10. This value can be initialized by sending and timing a series of predefined packets to the neighbor node 10.

During operation of the network 5, this value can be recalculated based on actual performance. For every target node 10, the NCM 20 on the node 10 sending the latency update sums the node latency (NLatencyTargetNode) and the PhysicalNetworkLatencyTargetNode for that target node 10 and then chooses the target node 10 that yields the lowest sum.

[0071] ServiceTimeOnQueue for a queue for which the latency update is sent is the time required for the NCM 20 on the node 10 to attempt to process all other queues on the before it comes back to the queue for which the latency update is sent. This period of time excludes the time used for processing the queue for which the latency update was sent. For each neighbor node 10 of a given node 10, the NCM 20 on the given node 10 maintains a list of queues, currently having EUM messages to send, that have this neighbor node 10 as a target node 10. In order to service queues fairly, the NCM 20 will cycle through all such queues in a round robin fashion. However, it will be apparent to one skilled in the art that it is also possible to designate the relative priority of queues, thus allowing the higher priority queues to be processed first. It is not the intention of the inventor to limit the order in which queues are processed to a particular round robin or priority queue algorithm.

[0072] As the NCM 20 on a node 10 calculating ServiceTimeOnQueue cycles through the list of queues created for each given target node 10, the NCM 20 will record the number of times it was able to send a message from a particular queue by incrementing a counter associated with that particular queue. The NCM 20 on the node calculating ServiceTimeOnQueue will only increment this counter if it is able to pass a message from this particular queue to a NTM associated with the given target node 10. The NCM 20 will only send a message from that particular queue to the given target node if it's node 10 has enough tokens for the target node 10 to broadcast the message for that particular queue. Further, the amount of data in that particular queue on the node 10 calculating ServiceTimeOnQueue must be greater than the NLatencyTargetNode of the given target node 10 minus the lowest

NLatencyTargetNode value of all other target nodes 10 for that queue. The NCM 20 on the node 10 calculating ServiceTimeOnQueue will also record how many messages it was able to send from all the queues 35 for the given target node 10.

[0073] The NCM 20 on the node 10 calculating ServiceTimeOnQueue will keep looping through this round robin process for a small number of cycles of the node's 10 operating system timer. For example, on a Windows 2000 system, the NCM 20 would loop through this process for 3-4 cycles of the low resolution system timer, taking approximately 45 to 60 milliseconds. For increased precision, the number of cycles may be increased. While the example provided applies to a node 10 using Windows 2000 as an operating system, it will be apparent to one skilled in the art that any operating system that applies a timing mechanism may be used. It is not the intention of the inventor to limit the scope of the invention to a particular operating system.

[0074] Once the required number of cycles of the operating system timer have elapsed for a target node 10, the NCM 20 on the node 10 calculating ServiceTimeOnQueue will record the total time in seconds used to go through the round-robin process for the number of required cycles of the node's 10 operating system timer. This time is stored in a variable TotalTimeForIterations. The NCM 20 will also record the total number of messages, in a variable TotalMessagesSentTargetNode, for all queues on the node 10 calculating ServiceTimeOnQueue, sent to the target node 10. In addition, the NCM 20 will record the number of messages that were sent from each individual queue for which the given target node 10 is designated as a target node 10. This is stored in a variable associated with each queue, TotalMessagesSentFromQueueTargetNode. The NCM 20 stores two iterations of the ServiceTimeOnQueue calculations, the one it is currently working on, and the last complete one. This allows calculation of the ServiceTimeOnQueue for the queue while continuing to gather new data for the next ServiceTimeOnQueue value. This value is only calculated when it is being sent as part of a calculation of node latency (NLatency) to reduce

computation overhead. To calculate ServiceTimeOnQueue for a given queue for the given target node 10, the NCM 20 uses the following ServiceTimeOnQueue formula:

ServiceTimeOnQueue =
(TotalMessagesSentTargetNode-
TotalMessagesSentFromQueueTargetNode) /
(TotalMessagesSentTargetNode * TotalTimeForIterations)

[0075] If there are multiple target nodes 10 for a queue, then ServiceTimeOnQueue is calculated using the above formula. Then, the following is used to derive ServiceTimeOnQueueAllTarget, which is the ServiceTimeOnQueue value for the queue for all target nodes 10:

ServiceTimeOnQueueAllTarget =
1/(1/[ServiceTimeOnQueue for target node X1] + 1/[Service time on queue for target node X2 + ...)

[0076] Using latency updates from all neighbor nodes 10, the NCM 20 on the node 10 that receives the latency updates uses the node latency (NLatency) formula provided above to calculate it's own latency for the queue and eventually transmits this to all neighbor nodes 10. Thus, the latency for each queue on each node 10 can be propagated to all nodes 10, as neighbor nodes 10, throughout the network 5. It should be noted that the BytesPerSecondSendRate, BytesInQueue, and ServiceTimeOnQueue values will not change no matter which target node 10 is chosen, thus making the NLatencyTargetNode and PhysicalNetworkLatencyTargetNode values the prime consideration.

[0077] For example, suppose a node 10 has two target nodes 10. The first target node 10 has a NLatencyTargetNode value of 3 seconds and a PhysicalNetworkLatencyTargetNode value of .5 seconds. The second target node 10 has an NLatencyTargetNode value of 2 seconds and a PhysicalNetworkLatencyTargetNode value of 2 seconds. For each target

node 10, the node 10 would add the NLatencyTargetNode and PhysicalNetworkLatencyTargetNode together. For the first target node 10, the sum would be 3.5 seconds. For the second target node 10, the sum would be 4 seconds. The node 10 would then choose the target node 10 with the minimum sum, namely the first target node 10 having a sum of 3.5. Suppose further that the service time on the queue, ServiceTimeOnQueue, is .1 seconds, that the node has 100 bytes in the queue, and that it's send rate is 200 bytes per second. Thus, assuming that all data has been sampled and calculated for MinOverTimePeriod, the node latency, as stored in NLatency, for the queue is 4.1, namely:

NLatency=
100 Bytes * 1 second/200 bytes BytesPerSecondSendRate + 0.1 ServiceTimeOnQueue + 3.5 MIN(NLatencyTargetNode + PhysicalNetworkLatencyTargetNode)

[0078] When attempting to compare node latencies to choose new target nodes 10, the NCM 20 will also apply the node latency formula to neighbor nodes 10 that are not target nodes 35. Thus, if the NCM 20 picks a neighbor node 10 as a new target node 10 for a queue, then the NCM 20 may store and generate a latency update for node latency that takes into account the new target node 10 for that queue. However, if the neighbor node 10 is not chosen as a target node 10, then the NCM 10 will only use the node latency as calculated with the existing target nodes 10.

[0079] For each queue on a node 10, the NCM 20 on that node 10 also maintains a variable, ACStatus, that marks the at capacity status for the queue. The at capacity status indicates whether a queue on that node 10 is sending or receiving EUM messages, via the NCM 20, at capacity for the queue. While at capacity status is tracked for all queues, at capacity status is above all used for terminating queues 35, as terminating queues 35 are the only modules used for carrying EUM data messages, which are comparatively larger in quantity and size than the other types of EUM messages, which are carried by information queues 35 and initiating queues 45.

[0080] A queue on a node 10 is considered at capacity if the node 10 is unable, by transmitting data to it's target nodes 10, to bring the amount of data in the queue down to a level wherein EUM messages can be transmitted from the queue almost as soon as they are received. For the purposes of the present invention, the ACStatus variable is set to true, indicating that a node 10 is at capacity, when the queue latency, or the number of seconds worth of data outstanding to be sent from that queue, exceeds the maximum value for all target node 10 node latencies minus the minimum value for all target node 10 node latencies for more than 5 time intervals, for example. If queue latency is stored in a variable QLatency and node latency for a target node 10 is stored again in a variable NLatencyTargetNode, his can be expressed as follows:

ACStatus = True, IF
QLatency > Max (NLatencyTargetNode of all target nodes) -
Min (NLatencyTargetNode of all target nodes)

Otherwise the node 10 is not at capacity and the ACStatus variable is set to false.

[0081] A time interval for the purposes of calculating at capacity status for the ACStatus variable is defined as the time required for every target node 10 able to send to send a certain number of messages (for instance, 10). The time interval for calculating at capacity status could also be set as a minimum time period or a maximum time period. For example, double the minimum granulation of a fast operating system timer could be used as a minimum time period. Alternatively, a maximum time period such as six seconds could be set. It is important, however, that enough time has elapsed during the time interval that the target nodes 10 have had the opportunity to bring the total amount of data in the queue to the lowest point they can. For example, if data is flowing in at 100 bytes every second, and flowing out at 500 bytes every five seconds, an absolute minimum time interval of 5 seconds would be required. The number and duration of time intervals enumerated above is provided for exemplary purposes. The actual number of time intervals will

depend on the number of target nodes 10 and the speed of data transmission. It is not the intention of the inventors to restrict the number or duration of the time periods or intervals to the examples provided.

[0082] Reference is now made to Figure 6, a flowchart of the steps for processing latency updates. As such, Figure 6 is an expanded view of step 65 of Figure 4. Commencing at step 85, the receiving node 10 receives the update containing the queue's queue label, node latency, and at capacity status from the sending neighbor node 10. Moving to step 90, the NCM 20 on the receiving node 10 verifies whether it has been informed of this queue label before. If not, this is a new queue for the receiving node 10 and, moving to the step 95, a new queue process is executed. During this new queue process, the NCM on the receiving node 10 processes the new queue and decides whether to propagate the new queue label to all neighbor nodes 10. Otherwise, at step 100, if the NCM 20 on the receiving node 10 has already been informed of the queue, i.e. the NCM 20 already has knowledge of the queue's queue label, the NCM 20 records the node latency and at capacity status for that queue for the sending neighbor node 10. Advancing to step 105, the NCM 20 on the receiving node 10 will use this information in a procedure for adjusting it's list of target nodes 10 for the queue. As will be explained, this adjustment may include adding or removing the sending neighbor node 10 or another neighbor node 10 to the list of target nodes 10 for the queue on the receiving node 10. The NCM 20 on the receiving node 10 node also uses the node latency information and at capacity status to calculate it's own node latency and at capacity status for the queue using the formulas for node latency (NLatency) and at capacity status (ACStatus) provided above. The NCM 20 on the receiving node 10 will then send this information to neighbor nodes 10 in latency updates. In this manner, at capacity status and latency information for a queue are updated and propagated throughout the network 5.

[0083] Reference is now made to Figure 7, a flowchart of the steps for processing a new queue. As such, Figure 7 is an expanded view of step 95 of

Figure 6. All operations for processing are queue are carried out by the NCM 20 on the receiving node 10 for a latency update. First, at step 110, the NCM 20 reads the node latency for the queue on the sending neighbor node 10. Next, at step 115, the NCM 20 verifies whether the node latency for the queue on the sending neighbor node is at infinity. As will be explained, when a node 10 has a latency of infinity for a queue, this indicates that a node 10 has no target nodes 10 for that queue. Should this be the case, moving to step 120, the NCM 20 will ignore the new queue label. Otherwise, proceeding to step 125, the NCM 20 creates a queue having the same queue label. A quota of tokens will also be negotiated. Moving to step 130, the NCM 20 records the node latency and at capacity status for the queue, associated with the sending neighbor node 10. Advancing to step 135, the NCM 20 designates the sending neighbor node 10 as a target node 10 on the list of target nodes 10 for the queue.

[0084] Moving now to step 140, the NCM 20 on the receiving node 10 for the latency update for the new queue informs the sending neighbor node 10 that the sending neighbor node 10 has been designated as a target node 10 for the new queue. This is done to avoid creation of an instant loop wherein two neighbor nodes 10, via their NCMs 20, are designated as target nodes 10 for each other for the same queue. Finally, at step 145, the NCM 20 calculates the node latency and at capacity status for the new queue on it's own node. The NCM 20 will send this information to all neighbor nodes 10 to inform them of the new queue in a latency update. At this stage, the NCM 20 also adds the neighbor nodes 10 to which it transmitted the new queue's queue label and accompanying node latency and at capacity status C to a list of nodes to which the NCM 20 has transmitted a latency update for the new queue. This ensures that the NCM 20 on the receiving neighbor node 10 maintains knowledge of the queues about which it has informed it's neighbor nodes 10. If the NCM 20 on any node 10 determines that it has not transmitted a latency update for a queue of which it is aware to a neighbor node 10, it will do so immediately to ensure that the existence of any new queue on the node 10 will be transmitted to all neighbor nodes 10. For

example, in the event that a new node 10 was added to the network 5, every neighbor node 10 for that new node 10 would send a latency update for every queue present on the neighbor node 10.

[0085] Reference is now made to Figure 8, a flowchart of the steps for updating the target nodes 10 for a known queue when receiving a latency update. Figure 8 is an expanded view of step 105 of Figure 6. Unless otherwise specified, all processing of the latency update is carried out by the NCM 20 on the receiving node 10 for a latency update. Commencing at step 150, the NCM 20 determines whether the node latency of the sending neighbor node 10 is infinity. Proceeding to step 155, if the node latency of the sending neighbor node 10 is not at infinity, then the NCM 20 will use the latency update to calculate the receiving node's 10 latency and capacity status for the queue when the NCM 20 sends it's next latency update for the queue. Moving to step 160, should the sending neighbor node 10 have a node latency of infinity for the queue, the NCM 20 will verify whether the sending node 10 is on the list of target nodes 10 for the queue. If not, the NCM 20 will proceed to step 155 and calculate the receiving node's 10 latency and at capacity status. Otherwise, progressing to step 165, the NCM 2 will remove the sending neighbor node 10 from the list of target nodes 10 for the queue. Next, at step 170, the NCM 20 will verify whether it has any target nodes 10 in it's list of target nodes 10 for the queue. If there is at least one target node 10 for the queue, the NCM 20 will proceed to step 155 and calculate the receiving node's 10 node latency and at capacity status for the next latency update. Otherwise, moving to step 175, the NCM 20 will set the node latency of the receiving node 10 to infinity for the queue. Then, at step 180, the NCM 20 will stop sending any EUM messages for the queue. Finally, at step 185, the NCM 20 will immediately send a latency update to all neighbor nodes 10 of the receiving node 10 to inform them that the node latency on the receiving node 10 has been set to infinity for the queue. Figure 8 represents the basic scenario applicable for updating target nodes 10. Other, more complex scenarios, such as adding additional target nodes 10

for data flow nodes 10 and reconstructing potential routes, are described subsequently.

[0086] Figure 9 is a flow chart of the steps for propagating a new queue and establishing potential routes for the new queue. As such, Figure 9 is an expanded view of step 70 of Figure 5. These steps are carried out by the NCM 20 on each node 10 involved. Initially, at step 190, a new queue is created by the NCM 20 for an EUM 30 on the origin node 10 for the queue. Next, at step 195, the origin node 10 transmits the queue label, node latency and at capacity status for the new queue to all neighbor nodes 10 in a latency update.

[0087] Proceeding to step 200, when the neighbor nodes 10 receive the latency update, they process the latency update's information. Thus, step 200 comprises an application of step 65 of Figure 5, shown in an expanded view in Figure 5. Further, since it is a new queue label being processed, step 95 for processing new queues of Figure 6, shown in an expanded view in Figure 7, is executed. Therefore, provided the node latency of the new queue is not infinity, each neighbor node 10 of the origin node 10 will create a queue having the same label as the new queue and will have the new queue's at capacity status and node latency information for the origin node 10. The NCMs 20 on the neighbor nodes 10 of the origin node 10 calculate their own target nodes 10, node latency and at capacity status for the new queue. Turning to step 205, the neighbor nodes 10 transmit latency updates with the new queue label, to all of their own neighbor nodes 10. Thus, step 65, and therefore step 95 for processing new queues is applied again. This process repeats continuously with each neighbor node 10 transmitting latency updates for the new queue. Since all nodes 10 in the network 5 have links to at least one neighbor node 10, knowledge of the new queue, as well as associated node latency and at capacity status is eventually propagated to each NCM 20 on each node 10 in the network 5. Further, since all nodes 10 will receive a latency update to inform them of the new queue, each node 10 will initially have a neighbor node 10 that is designated as a target node 10 for sending to

the new queue. Further, since step 95 for processing new queues is only executed by the NCM 20 if the queue label is new, this ensures that the NCM 20 of each node 10 chooses only the first neighbor node 10 to inform it of the new queue label as the initial target node 10. As each node 10 initially has only one target node 10 for the new queue and each successive target node 10 also has only one target node 10, each node 10 initially only has one potential route of target nodes 10 for the queue to the origin node 10.

[0088] Proceeding to step 210, when all nodes 10 have received latency updates for the new queue, potential routes to the origin node 10 for the new queue will have been established for all nodes 10 in the network 5. It should be noted, however, that there is no explicit test for determining whether the NCMs 20 on all nodes 10 have been informed of the new queue that stops the latency updates. Rather, every node 10 will continue to receive latency updates for the new queue and it's NCM 20 will use the latency updates to update the choice of target nodes 10. This is part of the step 80 for updating and maintaining data flow and potential routes, shown in Figure 5.

[0089] In the case of a very large number of queues present in the network 5, the NCM 20 may use a scheme to only immediately tell neighbor nodes 10 about queues that are most relevant. For example, the NCM 20 could be programmed to initially tell neighbor nodes 10 only about the queues having the highest amount of data, in the form of EUM messages, being transferred through them. Then, using bandwidth throttling, the node 10, via the NCM 20, could tell the neighbor nodes 10 about the remaining queues. Similarly, if a node 10 does not contain enough memory to store the queue labels, node latencies, or other data for every queue in the network 5, the NCM 20 can be programmed to ignore those queues it deems as un-important. The only side effect of this approach would be that would be that the nodes 10 that do not receive latency updates for a new queue would be unable to form a potential route or EEC to the origin node 10 of the new queue. These nodes 10 would not be usable for such potential routes or

EECs until they received a latency update concerning the new queue and had sufficient resources to process the latency update. It will be apparent to one skilled in the art that other methods for managing large numbers of queues may be possible. It is not the intention of the inventors to limit the methods for handling large numbers of queues to a specific method.

[0090] To better explain the steps for propagating a new queue and creating a potential route shown in Figure 9, an example of propagation of a new queue, using the network 5 shown in Figure 1, is provided in Table 6. For the purposes of the example, suppose that a new queue is created for an EUM 30 having a GUID of 123.X on node 10a. Suppose further that the node latency value, NLatency, for 123.X on node 10a is 0 and that the PhysicalNetworkLatencyTargetNode value for each node 10 is 1 second. No EUM messages are yet being transmitted in the new queue, thus making the amount of data in the queue 0. No node is at capacity, and therefore every node has an ACStatus value of false for the queue. Suppose further that there are no other queues in the network 5, making the ServiceTimeOnQueue value 0. Table 3 shows the resulting target nodes 10, node latency values, at capacity status, and potential routes from application of the steps for propagating a new queue.

Table 9

Node	Initial target node (first node to inform of queue label)	Node Latency (NLatency)	At Capacity (ACStatus)	Potential Route to origin node 10a (Left to right order)
10a	None	0	False	Not Applicable
10b	10a	1	False	10a
10c	10b	2	False	10b,10a
10d	10c	3	False	10c, 10b, 10a
10e	10f	4	False	10f, 10g,10b,10a

10f	10g	3	False	10g, 10b, 10a.
10g	10b	2	False	10b, 10a

[0091] As shown in Table 6, after the queue is first created on the origin node 10a for the EUM 30 at step 190, the NCM 20 on origin node 10a proceeds to step 195 and transmits a latency update to it's neighbor nodes 10, namely 10b. Proceeding to 200, 10b's NCM 20 processes the latency update, thus applying step 65 of Figure 5, and therefore step 95 of Figure 6, shown in an expanded view in Figure 7. Thus, the NCM 10 on node 10b calculates it's own node latency and at capacity status and designates 10a as a target node 10. Node 10b's NCM 20 then sends a latency update to the nodes 10 that are it's neighbor nodes 10c and 10g. Turning to step 205, the NCMs 20 on nodes 10c and 10g execute the latency update procedure, including the steps for processing a new queue shown in Figure 7, and choose their target nodes 10. For node 10c and 10g, the target node is node 10b. Node 10c and 10g then issue a latency update, via their NCMs 20, for the new queue to their neighbor nodes 10 at. For 10c, the neighbor nodes 10 are nodes 10b, 10f, and 10d. For node 10g, this is node 10b and 10f. For 10f, since 10g is the first node to inform 10g of the new queue, 10f's NCM 20 chooses 10g as the target node 10. The NCM 20 for 10d chooses 10c as the target node 10. On 10d and 10c, the NCM 20 informs their neighbor nodes 10 of the new queue with a latency update. For 10d, the neighbor nodes 10 are 10e and 10c. For 10f the neighbor nodes 10 are 10g, 10e, and 10c. Since 10e received the latency update from 10d, the NCM 20 on 10e picks 10d as it's target node 10. At this point, step 210 has been reached as all nodes 10 have received a latency update for the new queue. Thus, a potential route of target nodes 10 to the origin node 10a for the new queue has been established for each node 10.

[0092] As for node latency values, since there are no messages yet being transmitted for the queue and there are no other queues in the network 10, this means that MinOverTimePeriod, BytesPerSecondSendRate, and

ServiceTimeOnQueue all have values of 0. Further, since there is initially only one target node 10, namely the first neighbor node to inform each node 10 of the new queue, the node latency for the new queue is initially the sum of the NLatencyTargetNode for the target node 10 added to the PhysicalNetworkLatencyTargetNode value for the target node. Thus, for node 10b, the node latency value is 1, namely the NLatencyTargetNode value for 10a, namely 0, plus the PhysicalNetworkLatencyTargetNode to reach 10a, i.e. 1. The node latency values for the other nodes 10 are calculated in a similar fashion.

[0093] Since the NCM 20 on each node 10 only has knowledge of that node's 10 neighbor nodes 10, at no point does any NCM 20 on any node 10 attempt to build a global network map, or have any knowledge of the network 5 as a whole. At the same time, the NCM 20 on each node 10 has knowledge of every queue associated with every EUM 30 and a potential route of target nodes 10 for every queue to each queue's origin node 10. Thus, EUM messages can be sent from an EUM 30 on one node 10 to an EUM 30 on any other node 10 without maintenance of any global network topology information. This allows the network 5 to function more efficiently for the purposes of rapid data transmission and reduction of overhead.

[0094] Reference is now made to Figure 10, a flowchart of the steps for establishing an EEC and data flow. As such, Figure 9 is an expanded view of step 75 of Figure 5. EECs are established by NCMs 20 at the request of EUMs 30. In order to establish an EEC from an initiating EUM 30 on an initiating node 10 to a terminating EUM 30 on a terminating node 10, commencing at step 215, the terminating EUM 30 on the terminating node 10 must have first created an information queue 35. Further, the information queue 35 must have been propagated throughout the network 5 to the initiating node 10 by executing step 65 of Figure 5, shown in an expanded view in Figure 9. Proceeding to 220, the initiating EUM 30 on the initiating node 10 requests that the initiating node's 10 NCM 20 establish an EEC for the EUM 30 associated with the information queue 35. Upon receiving that

request, NCM 20 on the initiating node 10 creates an initiating queue 45, associated with the initiating EUM 30 requesting the EEC. The knowledge of the initiating queue 45 is propagated throughout the system using procedure by executing step 65 of Figure 5, shown in an expanded view in Figure 9. Thus, potential routes of target nodes 10 for all other nodes 10, including the terminating node 10, to the initiating node 10 for the initiating queue 45 is created.

[0095] Moving to step 225, the NCM 20 on the initiating node 10 sends an EEC message to the terminating node's 10 information queue 35 requesting an EEC by which the initiating EUM 30 can send messages to the terminating EUM 30 on the terminating node 10 and asking that a reply be sent to the initiating node's 10 initiating queue 45. The EEC request message also contains the queue label associated with the new initiating queue 45 on the initiating node 10. The EEC message for the request is transmitted through the potential route of target nodes 10 for the information queue 35.

[0096] Moving to step 230, when the EEC request reaches the terminating node 10, the NCM 20 on the terminating node 10 informs the terminating EUM 30 of the EEC message requesting the EEC by placing the message in the information queue 35. The NCM 20 on the terminating node then creates a new terminating queue 40 for that specific EEC for the initiating EUM 30 and allocates a buffer for the terminating queue 40 with a fixed buffer size for reordering out of order EUM messages sent from the initiating node 10 and received on the terminating node 10 over the EEC. Knowledge of the terminating queue 40 is also propagated throughout the system by executing step 65 of Figure 5, shown in an expanded view in Figure 9. Thus, initial potential routes, including a potential route from the initiating node 10, are established for the terminating queue 40 on the terminating node 10.

[0097] At this point the involvement of the information queue 35 in establishing the EEC ends. It is the terminating queue 40 and the initiating queue 45 that form the basis for the EEC. The information queue 35 may continue to be used for establishing other EECs to the terminating EUM 30 for

the purposes associated with information queue 35. Subsequent terminating queues 40 and initiating queues 45 will be being created for each new EEC.

[0098] Turning now to step 260, once the terminating queue 40 is propagated, the initial data flow for the terminating queue 40 of the EEC may be established. Since, each node 10 initially only chooses one target node 10 for a queue, the data flow initially only consists of one end-to-end data path. The data flow for the terminating queue 40 on an EEC is the potential route to the terminating queue 40 on the terminating node 10 from the initiating node 10. To mark the initial data flow, the NCM 20 on the initiating node 10 sets a data flow flag to true for the terminating queue 40 and then informs it's target nodes 10 that they are part of the data flow. In turn, the target nodes 10, via their NCMs 20, set their data flow flags to true and inform their own target nodes 10 for the terminating queue 40. This process continues until all target nodes 10 on the path of target nodes 10 for the terminating queue 40 leading from the initiating node 10 to the terminating node 10 for the EEC have marked their data flow flags to true, thus establishing the data flow. The target nodes 10 with their data flow flags set to true are now data flow nodes 10 for the terminating queue 40 of the EEC. The NCMs 20 on the nodes 10 that are not data flow nodes 10 nonetheless maintain their records of target nodes 10, as the potential routes from these nodes 10 to the terminating node 10 for the terminating queue 40 may eventually become part of the data flow.

[0099] Moving to step 240, the NCM 20 on the terminating node 10 then sends an EEC message back to the initiating queue 45 on the initiating node 10 along the potential route established from the terminating node for the initiating queue 45. This message contains the queue label for the terminating queue 40, as well as the terminating queue buffer size. Proceeding to step 245, once this message reaches the initiating node 10, the initiating node 10 creates a initiating queue 45 buffer for re-ordering acknowledgement messages that are received out of order. The size of the initiating queue buffer is controlled by the initiating node 10 and it will always be equal or less than the terminating queue buffer size.

[00100] Finally, at step 250, the initiating EUM 30 on the initiating node 10 can commence transmitting EUM data messages and EEC messages to the terminating queue 40. As the terminating queue 40 on the terminating node 10 receives data, the terminating EUM 30 on the terminating node 10 sends acknowledgment messages to the initiating queue 45 on the initiating node 10. The EEC messages and EUM data messages from the initiating node 10 are sent along the path or paths of target nodes 10 in the data flow, i.e. the data flow nodes 10 for the terminating queue 40. The acknowledgement messages and EEC messages for the initiating queue 45 are sent on the potential route for the initiating queue 45 from the terminating node 10 to the initiating node 10. If the initiating EUM 30 does not receive an acknowledgement message for an EUM data message sent from the initiating node 10 after a fixed amount of time, then the initiating EUM 30 will have the NCM 20 on the initiating node 10 resend that acknowledgement message in the terminating queue 40. The quantity of EUM data messages sent by the initiating EUM 30 on the initiating node 10 through the network 5 to the terminating queue 40 on the terminating node 10 may not exceed the terminating queue buffer size. The NCM 20 on the terminating node 10 can increase or decrease the terminating queue buffer size at any time and will inform the initiating node 10 of this by sending an EEC message to the initiating queue 45. The terminating queue buffer size could be changed, for example, if there was more or less bandwidth available for the EEC.

[00101] Acknowledgements messages can be grouped together to conserve bandwidth. This task is effected by the NCM 20 on each target node 10 along the potential route for the initiating queue 45 from the terminating node 10 to the initiating node 10. For example the acknowledgement of messages 10-35 and 36-50 sent from an initiating node 10 could be grouped as a single acknowledgement message sent form the terminating node 10 for messages 10-50 of the initiating node 10. This allows multiple acknowledgements to be represented in a single acknowledgement message. Should the initiating queue 45 or the initiating node 10 become inaccessible to the network 5, the NCM 20 on the terminating node 10 will

remove the terminating queue 40 and de-allocate all buffers associated with the EEC. Similarly, if the NCM 20 on the initiating node becomes aware that the terminating queue 40 is no longer visible, then the NCM 20 on the initiating node 10 will deallocate the initiating queue 45. This will only occur if all possible paths of target nodes 10 for the data flow between the initiating node 10 and the terminating node 10 for the terminating queue 40 are removed, all possible paths of target nodes 10 for the potential route to the initiating queue 45 are removed, or one or both of the terminating node 10 or initiating node 10 terminates the EEC.

[00102] Reference is now made to Figure 11, a flowchart of the steps for maintaining and updating data flow and potential routes. Figure 11 is an expanded view of step 80 of Figure 5. Once the EEC connection and it's initial data flow are established, the data flow for the EEC and potential routes will be maintained and optimized, including adding new, multiple target nodes 10 for data flow nodes 10, based on node latency and at capacity status. Nodes 10 may also be removed from the data flow and potential routes. As with establishment of data flow and potential routes, the actual calculations for data flow and potential route maintenance are done on a node-by-node basis by the NCM 20, rather than on an end-to-end basis, even though they result in the creation of end-to-end potential routes and data flow for EECs. The NCM 20 on each node 10 maintains it's list of target nodes 10. For data flow purposes, the NCM 20 on each data flow node 10 updates it's data flow flag for each terminating queue 40 and then informs target nodes 10 that they are, or they are not, in the data flow. Data flow and potential route maintenance involves four concurrent steps:

(a) Step 255 for maintaining at least one target node 10 for each queue on each node 10. Step 285 ensures that there is at least one path for potential routes. It also ensures maintenance of the data flow when a data flow node 10 has only one target node 10.

(b) Step 260 for optimizing data flow based on at capacity status.

(c) Step 265 for removing unused target nodes 10 from the data flow.

(d) Step 270 for optimizing potential routes for nodes 10 not in the data flow.

[00103] Reference is now made to Figure 12, a flowchart of the steps for maintaining at least one target node 10 for each queue on each node 10. As such, Figure 11 is an expanded view of step 255 of Figure 11. All steps are carried out by the NCMs 20 of the nodes 10 involved. Maintaining at least one target node 10 for a queue on every node 20 is essential for maintaining potential routes from all nodes 10 to the origin node 10 for the queue. Also, for an EEC, every data flow node 10 must have at least one target node 10 to maintain the data flow to the terminating node 10 for the terminating queue 40. A node's 10 only path on the data flow or potential route to any queue, whether information queue 35, terminating queue 40, or initiating queue 45 is rendered invalid by a loss of all target nodes 10. This may occur in three circumstances:

(a) A queue is removed;

(b) The origin node 10, whether a terminating node or initiating node 10, for a queue is shut down unexpectedly or removed unexpectedly; or

(c) A node 10 that has only one target node 10 for a queue is shut down or otherwise disconnected from the node 10 that it was using as it's only target node 10 for the queue.

[00104] Beginning at step 275, the node 10 loses all target nodes 10, rendering it's potential route or only path for data flow to the origin node 10 for a queue invalid, as described in the three situations above. Proceeding to step 280, the NCM 20 of the node 10 will query all neighbor nodes 10 to see if they selected the node 10 as a target node 10 to avoid creating an instant loop. Turning to step 285, if there are neighbor nodes 10 available that do not create an instant loop, then the NCM 20 on the node 10 will conduct a loop GUID test for all such neighbor nodes 10 to attempt to find a neighbor node

10 that can be used as a target node 10 without forming a loop elsewhere. In executing the Loop GUID test, the NCM 20 on the node circulates loop test GUIDs around the path of a target nodes 10 emanating from the neighbor node 10 being tested. If the loop test GUID is returned, then the NCM 20 can conclude that selection of the neighbor node 10 being tested as a target node 10 would create a loop. Proceeding to 290, the NCM 20 evaluates the results of the loop GUID test from 285 to determine if any neighbor nodes 10 exist for selection as the new target node 10 that do not create a loop. These are potential target nodes 10. From these potential target nodes 10, the NCM 20 will pick a given potential target node 10 and will query the potential target node's 10 NCM 20 to ensure that the potential target node 10 has itself not picked the node 10 as a target node 10. This is done to determine, one last time, whether an instant loop will be created if the NCM 20 on the node 10 chooses the potential target node 10.

[00105] If an instant loop is created, the NCM 20 will test another potential target node 10 for instant loops 10. For each potential target node 10 that the NCM 20 examines, the NCM 20 will deactivate any traces of the loop GUID's data structures used in the loop GUID test. If all potential target nodes 10 identified by the loop GUID test at 285 create instant loops, then the NCM 20 will conclude there are no non-looping options for target nodes 10. However, moving to step 295, if there are one or more potential target nodes 10 that do not create an instant loop, the NCM 20 selects one of the potential target nodes 10 as the new target node 10.

[00106] Proceeding to step 300, if all potential target nodes 10 create loops, the NCM 20 on the node 10 will set the node latency, NLatency, for the queue to infinity and immediately send a latency update to propagate the new node latency information to all neighbor nodes 10. Upon receiving the latency update, any receiving neighbor node 10 that has designated the node 10 having a node latency of infinity as target node 10 will stop sending EUM messages to the node 10 having the node latency of infinity. The NCM 20 on the neighbor node 10 will also remove the node 10 having the node latency of

infinity from the list of target nodes 10 for the queue. If, as a result of removing the node 10 having the node latency of infinity as a target node 10, the neighbor node 10 is left with no target nodes 10 for the queue, the NCM 20 on the neighbor node 10 will set it's own node latency to infinity for the queue. The NCM 20 on the neighbor node 10, will, in turn, immediately transmit it's new node latency for the queue to it's own neighbor nodes 10. As all nodes 10, via their NCMs 20, send latency updates to their neighbor nodes 10, this process will continue until the node latency for the queue of every node 10 that can be set to infinity is, in fact, set to infinity.

[00107] Proceeding to step 305, the NCM 20 of any node 10 that has a node latency of infinity will maintain EUM messages and will wait for a predetermined and fixed target node reestablishment time period. At the end of the target node reestablishment time period, at step 310, the NCM 20 will again verify whether there are any neighbor nodes 10 for which the node latency is not at infinity. These are potential target nodes 10. If so, returning to step 295, the NCM 20 instantly selects a new target node 10 from the potential target nodes 10. The selection at 295 is governed by the following rules:

(a) If there is only one non-looping potential target node 10 available that is not at infinity, the NCM 20 will choose that potential target node 10 as the target node 10 regardless of whether the potential target node 10 is at capacity or not.

(b) If there is more than one non-looping potential target node 10 that does not have a node latency of infinity, then the NCM 20 chooses the potential target node 10 with the lowest node latency that is not at capacity as the target node 10 for the queue.

(c) If all non-looping potential target nodes 10 that do not have a node latency of infinity are at capacity, the NCM 20 chooses the potential target node 10 with the lowest node latency as the target node 10 for the queue

[00108] Proceeding to step 315, the NCM 20 on the node 10 that has chosen the new target node 10 instantly recalculates it's new node latency and immediately sends a latency update to all neighbor nodes 10. This

ensures that, for a node 10 that had a node latency of infinity and has just chosen a new target node 10, that node's 10 new, non-infinity, node latency and it's at capacity status are instantly transmitted to the neighbor nodes 10. Upon receiving the latency update, the NCMs 20 of any neighbor nodes 10 with a node latency at infinity may each attempt to select a new target node 10 from their neighbor nodes 10, such as the node 10 that has just chosen a new target node 10. The rules for selection are those set out for step 295. Thus, the NCM 20 of any node 10 with a node latency at infinity will select the first neighbor node 10 that provides it with a non-infinite node latency as that node's 10 new target node 10. This process is repeated for each neighbor node 10 until all nodes 10 once again have a target node 10 to point to the queue, thus reestablishing potential routes and data flow.

[00109] Proceeding to step 320, if, at , a node 10 is incapable of locating a neighbor node 10 that does not have a node latency of infinity, it's NCM 20 will verify whether a predetermined time out period has expired. Moving to step 325, if the timeout period has not expired, the NCM 20 will once again check for a neighbor node 10 that does not have a node latency of infinity. If there is such a neighbor node 10, proceeding again to step 295, the NCM 20 selects that neighbor node 10 as the new target node 10. If the timeout period has expired and the NCM 20 has not located a neighbor node 10 without a node latency at infinity, the NCM 20 concludes that there is no valid path for the queue. In such a case, moving to step 330, the NCM 20 then deletes all EUM messages and data for the queue, including the NCM's 20 own knowledge of the queue.

[00110] To further aid the reader in understanding the steps for maintaining at least one target node 10 for each queue on each node 10, shown in Figure 11 as an expanded view of step 255, an example is provided using Table 10, Table 11 and the network 5 shown in Figure 1. Suppose that a terminating queue 40 has been created for an EUM 30 on node 10a, making 10a the terminating node 10 for the terminating queue 40, and propagated. Suppose further that potential routes have been established according to

Table 7 and that an EEC, and therefore data flow, between 10f, the initiating node, and 10a has been established.

Table 10

Node	Target node: Terminating Queue	At Capacity (ACStatus)	Potential Route/Data flow to terminating node 10a (Left to Right Order)
10a	Not applicable	False	Not Applicable
10b	10a	False	10a
10c	10b	False	10b,10a
10d	10c	False	10c, 10b,10a
10e	10d	False	10d, 10g,10b,10a
10f	10e	False	10e, 10d, 10c, 10b, 10a.
10g	10b	False	10b, 10a

[00111] If data link 15b were to become inoperative, then node 10c would lose it's target node 10. Further, selection of 10f as a target node for 10c would create a loop consisting of 10c, 10f, 10e, 10d. Thus the GUID loop test at step 285 would fail and the NCM 20 on node 10c would set the node latency for the terminating queue 40 to infinity and inform it's neighbor nodes, 10d and 10f, with a latency update. As 10d would have no more non-looping target nodes 10, it's node latency would also be set to infinity and it's NCM would alert neighbor nodes 10e and 10f. Nodes 10e and 10f would also lose their target nodes 10, 10e and 10f, and their node latency would also be set to infinity. At this point, nodes 10c, 10d, 10e, and 10f all have node latencies of infinity and have no target node 10 for the terminating queue 40. Nodes 10g

and 10b continue to have valid target nodes 10. Thus, all required nodes 10 have been set to infinity. Eventually, after it's target node reestablishment period, the NCM 20 on node 10f will attempt to pick a neighbor node 10 that does not have a latency of infinity as a new target node 10. The NCM 20 on node 10f will discover that node 10g is available and will choose 10g as it's new target node 10. The NCM 20 on node 10f will then recalculate it's node's new, non-infinite latency and transmit this to neighbor nodes 10c and 10e in a latency update. The NCMs 20 on 10c and 10e will in turn choose 10f as a new target node 10 and recalculate the new node latencies for 10c and 10e. When the NCMs 20 for nodes 10e and 10c transmit their latency updates, 10d will be informed of the new non-infinity node latencies for 10c and 10e. Suppose that the target node reestablishment period for node 10d has expired and that node 10e's latency update reaches node 10d first. Thus, node 10d's NCM 20 chooses node 10e as a new target node 10. At this point, all nodes 10 once again have target nodes 10 and potential routes to the terminating queue 40 on node 10a. Since node 10f is an initiating node 45, it's target node will always be in the data flow. As subsequent target node's 10 for 10f's target node 10 create a path of target node's back to 10a, their data flow flags are set to true. Thus the data flow to 10a for the terminating queue 40 for the EEC from 10f to 10a is also re-established. The new values for the target nodes 10, potential routes, and data flow would be as shown in Table 5:

Table 11

Node	Target node	At Capacity Status (ACStatus)	Potential Route/ Data flow to terminating node 10a (left to right order)
10a	Not applicable	False	Not Applicable
10b	10a	False	10a

10c	10f	False	10f,10g,10b,10a
10d	10e	False	10e, 10f, 10g, 10b, 10a
10e	10f	False	10f, 10g,10b,10a
10f	10g	False	10g,10b, 10a.
10g	10b	False	10b, 10a

[00112] As one skilled in the art will appreciate, this approach allows the present invention to maintain and rebuild potential routes and paths for data flow on an adaptive basis. This provides robustness and reliability for the network 5, once again without having to maintain knowledge of the entire network topography. Thus, while the present invention increases efficiency for data transmission with reduced overheard, it in no way sacrifices reliability and robustness. In fact, reliability and robustness are made less costly since it is not necessary to maintain any record of the global topology of the network 5. Nodes 10 only maintain knowledge, via their NCMs 20, of queue labels and node latency, token availability, data flow and at capacity status for their neighbor nodes 10.

[00113] Reference is now made to Figure 13, a flowchart for the steps of the loop GUID test. As such, Figure 13 is an expanded view of step 285 of Figure 11. The steps shown in then expanded view in Figure 13 are also executed 13 by the NCMs 20 on data flow nodes 10 for testing for loops. All steps shown in Figure 13 are carried out by the NCM 20 of each node 10 involved.

[00114] Initially, at step 335, the NCM 20 of the node 10 testing for the loop creates a unique loop GUID message for the neighbor node 10 that it wishes to test. A loop GUID message is composed of the loop GUID, the queue label for the queue concerned, and a loop GUID flag indicating whether the node 10 receiving the loop GUID message should remember or forget the loop GUID.

[00115] The loop GUID message is sent by the NCM 20 to the neighbor node 10 to be tested and propagated along the route of target nodes 10 for the neighbor node 10. In other words, the NCM 20 of the neighbor node 10 forwards the loop GUID message to all of it's target nodes 10 for the queue. These target nodes 10 in turn forward the loop GUID message to each of their respective target nodes 10 for the queue. As such, the loop GUID message is eventually passed through the entire path of target nodes 10 for the queue leading from the target node 10 of the neighbor node 10 being tested.

[00116] When the NCM 20 on any node 10 receives a loop GUID message, the NCM 20 will preempt normal message communications and send the loop GUID message as soon as possible. If the NCM 20 has already seen and processed a loop GUID message, the NCM 20 will ignore it. The NCM 20 on each node 10 that receives a loop GUID message checks the loop GUID flag to see whether it is set to remember or forget. If the loop GUID flag is set to remember, the NCM 20 records the GUID name and instructs all of the node's target nodes for the queue to remember the loop GUID in the loop GUID message. The NCM 20 also maintains a list of any neighbor nodes 10 from which it received a loop GUID message with the loop GUID flag set to remember. However, if all the nodes 10 from which an NCM 20 on a node 10 received loop GUID messages for a queue with the loop GUID flag set to remember subsequently send a loop GUID message for that queue with the loop GUID flag set to forget, or these nodes cease to be target nodes 10 for the queue, or the target nodes are disconnected, then the NCM 20 will deactivate that loop GUID message. An NCM 20 will deactivate a loop GUID message by setting the loop GUID flag to forget and transmitting the loop GUID message to all target nodes 10 for the queue involved.

[00117] Proceeding to step 340, the NCM 20 on the node 10 that initiated the loop GUID test waits for a minimum fixed amount of time (1 second for example) plus some random interval while the loop GUID message is propagated through the path of target nodes 10 from the neighbor node 10 being tested. Proceeding to step 345, the NCM 20 on the node 10 that

initiated the loop GUID test examines whether a loop GUID message has returned to it. If so, proceeding to step 350, the NCM 20 on the node 10 that initiated the loop test concludes that a loop exists for the neighbor node 10 tested if that neighbor node 10 is chosen as a target node 10. Otherwise, moving to step 355, the NCM 20 concludes that no loop would be created by choosing the neighbor node 10 tested as a target node 10. At this point, the NCM 20 may also send out a loop GUID message for the loop GUID message sent previously, with the loop GUID flag set to forget, so as to ensure that the loop GUID message is deactivated.

[00118] If, at the end of the loop GUID test on two neighbor nodes 10, the NCM 20 on each neighbor node 10 selects the other neighbor node 10 as a target node 10 at exactly the same time, the NCMs 20 on each neighbor node 10 will instantly switch back to the respective previous target node 10 and retry the process of finding additional target nodes 10. Since the loop GUID test includes a random interval, the likelihood of the two neighbor nodes 10 again selecting each other declines at each iteration.

[00119] Reference is now made to Figure 14, a flowchart of the steps for optimizing the data flow in an EEC. As such, Figure 14 is an expanded view of step 260 in Figure 11. When optimizing data flow, a new data flow node 10 is added to the data flow for an EEC. All steps shown in Figure 14 are carried out by the NCM. A new node 10 can be added to the data flow, thus becoming a data flow node 10, only by the NCM 20 on another node that is already a data flow node 10. Only data flow nodes 10 can have multiple target nodes 10 and, therefore, multiple paths for the terminating queue 40. The steps for adding new nodes 10 as target nodes 10 for a data flow node 10 is different when the data flow node 10 is at capacity than when the data flow node 10 is not at capacity 10. In both cases, nonetheless, the NCM 20 will attempt to find neighbor nodes 10 to add as new target nodes 10, and therefore new data flow nodes 10, that satisfy the following two criteria:

(a) The neighbor node 10 is not at capacity; and

(b) The addition of the neighbor node 10 as a target node 10, and therefore dataflow node 10, will not create a loop of target nodes 10.

[00120] Initially, at 360, the NCM 20 on a data flow node 10 calculates the at capacity status, stored in variable ACStatus, of the data flow node 10 for the terminating queue 40. If the data flow node 10 is at capacity, the next latency update will spread this information to all of the data flow node's 10 neighbor nodes 10. In addition, any node 10, data flow node 10 or not, will be set at capacity for a queue by it's NCM 20 if all of it's target nodes 10 are at capacity.

[00121] Proceeding to step 365, the NCM 20 on the data flow node 10 will examine the data flow node's 10 at capacity status for the terminating queue 40 to see if it is at capacity 10. If the data flow node 10 is at capacity 10, moving to step 370, the data flow node 10 will examine the node latency information and at capacity status for all neighbor nodes 10 and will identify all potential target nodes 10. All neighbor nodes 10 that have not already been designated as target nodes 10 and are not at capacity are potential target nodes 10. The NCM 20 on the data flow node 10 will also verify whether each potential target node 10 has already chosen the data flow node 10 as a target node 10 for the terminating queue 40. If so, this would create an instant loop and that neighbor node 10 is eliminated as a potential target node 10. If there are no potential target nodes 10, the NCM 20 will return to step 360 and recalculate it's node latency and at capacity status.

[00122] Turning now to step 375, for each remaining potential target node 10, the NCM 20 on the data flow node 10 at capacity will perform a GUID loop test to determine if the potential target node 10 will create a loop if actually chosen as a target node 10. The loop GUID test at step 375 involves the same steps performed for the loop GUID test at step 285 of Figure 12, an expanded view of which is shown in Figure 13.

[00123] Moving on to step 380, based on the results of the GUID loop test, the NCM 20 on the data flow node 10 at capacity determines whether

there are any non-looping potential new target nodes 10 available. If all potential new target nodes 10 are determined to create loops, the NCM 20 on the data flow node 10 at capacity will, at step 385, verify whether the data flow node 10 is, in fact, still at capacity. If not, the NCM 20 proceeds to step 390 and no new nodes 10 are added to the data flow as target nodes 10 for the data flow node 10. Instead, the NCM 20 on the data flow node 10 simply recalculates the data flow node's 10 node latency. The NCM 20 then transmits this data, along with the new, not at capacity, value for the data flow node's 10 at capacity status in the next latency update. If the data flow node 10 is still at capacity, the NCM 20 returns to step 375 and maintains the loop GUID test.

[00124] Proceeding to step 395, if the NCM 20 on the data flow node 10 at capacity 10 determines there is a non-looping potential target node 10 available, it will pick the lowest node latency potential target node 10 that is not at capacity and remove the loop test GUID message associated with that potential target node 10. The NCM 20 will also verify whether that potential target node 10 has itself chosen the data flow node 10 as a target node 10. This is to ensure, one last time, that an instant loop is not created. If an instant loop is created, the NCM 20 discards that potential target node 10 and returns to step 380 to verify whether the GUID loop test identified other potential target nodes 10. If so, the NCM 20 will again pick the lowest latency potential target node 10 that is not at capacity and test for an instant loop at step 395. If there are no potential new target nodes 10 left based on the results of the loop GUID test, the NCM 20 will again verify whether the data flow node 25 is still at capacity at step 385. If so, the NCM 20 will maintain the loop test GUIDs for the GUID loop test by returning to step 375. Otherwise, returning to step 390, the NCM 20 will not add any new target nodes 10 for the terminating queue 40.

[00125] Moving now to step 400, should the potential target node 10 not create an instant loop at step 395, the NCM 20 on the data flow node 10 at capacity will definitively add the potential new target node to it's list of target

nodes 10 and set it's node's at capacity status to false. This new at capacity status is propagated to all neighbor nodes 10 during the next latency update. Further, the target node 10 just chosen is informed of it's new status as a data flow node 10 and the target node's 10 NCM 20 sets the data flow flag to true for the terminating queue 40. Since the new target node 10 already has a designated chain of target nodes 10 to the terminating node 10, the addition of the target node 10 results in all of it's successive target node's 10 data flow flags also being set to true. Therefore, the successive target node's of the target node 10 just chosen by the NCM 20 on the data flow node 10 also become data flow nodes 10.. Thus, the designation of the new target node 10, which becomes a data flow node 10, adds a new path for the EEC's data flow.

[00126] Returning now to step 365, in the event that the data flow node 10 is not at capacity, it's NCM 20 will still attempt to seek out a lower latency target node 10, and therefore lower latency path, to the terminating node 10 for the terminating queue 40. However, the criteria used are different. Proceeding to step 405, when a data flow node 10 is not at capacity and receives a latency update for a terminating queue 40 from a neighbor node 10, the data flow node's 10 NCM 20 tests the node latency of the neighbor node 10 to determine whether this node latency is lower than the highest node latency for any of the receiving data flow node's 10 current target nodes 10. If not, then returning to step 390, the neighbor node 10 that sent the latency update will not be added to the list of target nodes 10. Otherwise, returning to step 370, the NCM 20 on the data flow node 10 that receives the latency update designates the neighbor node 10 that sent the latency update as a potential target node 10 for the terminating queue 40. The NCM 20 tests to see if selecting the neighbor node 10 would create an instant loop. Proceeding again to step 375, the potential new target node 10 is tested for loops using the same loop GUID test as when a data flow node 10 in the data flow is at capacity. However, since the data flow node 10 is not at capacity, the NCM 20 proceeds directly to step 390 from step 38 if a loop is identified at step 380. In this case, the neighbor node 10 is not added as a target node

10. Further, when the data flow node 10 is not at capacity, there is only one potential target node 10, namely the neighbor node 10 that sent the latency update. If the neighbor node 10 that sent the latency update does not create a loop, as verified at step 380, the NCM 20 will proceed to steps 395 and 400, as described previously.

[00127] To facilitate comprehension of the steps for adding a new target node to the data flow, shown in Figure 14 as an expanded view of step 260 of Figure 11, an example is provided. The example refers to the network 5 shown in Figure 1 and Tables 12, 13 and 14. Suppose that the network 5 is in the state indicated by Table 12 and that an EEC has been established between node 10d and node 10a, with node 10a being the terminating node 10 and node 10d being the initiating node 10. Thus the current data flow of target nodes 10 as data flow nodes 10 for the EEC is 10d, 10c,10b, 10a.

Table 12

Node	List of target nodes	At Capacity (ACStatus)	Potential Route/ Data flow (Left to Right)
10a	None	False	Not Applicable
10b	10a	False	10a
10c	10b	False	10b,10a
10d	10c	False	10c, 10b, 10a
10e	10f	False	10f, 10g,10b,10a
10f	10g	False	10g, 10b, 10a.
10g	10b	False	10b, 10a

[00128] Suppose data flow node 10c determines that it is at capacity. When node 10c sends out it's next latency update, all target nodes 10 for

node 10d will also be at capacity, thus putting node 10d at capacity. This puts the state of the network 5 as set out in Table 13.

Table 13

Node	List of target nodes	At capacity (AC)	Potential Route/Data flow to 10a (Left to Right)
10a	None	False	Not Applicable
10b	10a	False	10a
10c	10b	True	10b,10a
10d	10c	True	10c, 10b, 10a
10e	10f	False	10f, 10g,10b,10a
10f	10g	False	10g, 10b, 10a.
10g	10b	False	10b, 10a

[00129] Since 10c and 10d are data flow nodes 10 at capacity, the NCMs 20 of both nodes 10 will attempt to add new nodes 10 to the data flow. For purposes of simplicity, suppose that node 10c will complete the procedure 405 before node 10d. The NCM 20 on node 10c will seek out neighbor nodes 10 as potential new target nodes 10. This will result in the selection of node 10f as a potential new target node 10. The NCM 20 on node 10c will execute the GUID loop test, which will reveal that node 10f does not create a loop. Nor does selection of node 10f create an instant loop. Thus, data flow node 10c adds 10f to it's list of target nodes 10 and node 10c's at capacity status is set to false. Data flow node 10c will inform 10f that 10f is now part of the data flow and 10f will set it's data flow flag to true. This will in turn cause 10g's data flow flag to be set to true when it receives it's next latency update from 10f.

Finally, the NCM on 10g will inform 10a that it is in the data flow when 10g sends it next latency update.

[00130] Concurrently, data flow node 10d will attempt to test neighbor node 10e as a potential new target node 10. Since selecting 10e as a new target node 10 will also not create any loops, 10e is added to data flow node 10d's list of target nodes 10 and placed in data flow node 10d's data flow. Data flow node 10d's NCM 20 will also inform 10e that 10e is in the data flow, and this will cause 10e's target nodes to be informed that they too are in the data flow, with their data flow flags set to true. This information will be propagated to all subsequent target nodes 10, namely 10f, 10g, 10b and 10a. Thus, at this point, the state of the network 5 will be as set out in Table 14.

Table 14

Node	List of target nodes	At capacity status (ACStatus)	Potential Route/Data flow (Left to Right)
10a	None	False	Not Applicable
10b	10a	False	10a
10c	10b, 10f	False	10b,10a 10f,10g,10b,10a
10d	10e, 10c	False	10c, 10b, 10a 10e,10f,10g,10b,10a 10c,10f,10g,10b,10a
10e	10f	False	10f, 10g,10b,10a
10f	10g	False	10g, 10b, 10a.
10g	10b	False	10b,10a

[00131] Since 10f has been added as a target node 10 for data flow node 10c, the path of 10c, 10f, and 10a becomes part of the data flow from 10c. Since node 10e has been added as a target node 10 for data flow node 10d, the path provided by nodes 10e, 10f, 10g, 10b, and 10a has been added as part of the data flow for node 10d. Finally, since 10c is already as target node 10, and part of the data flow for node 10d, the new path provided by 10c, 10f, 10g, 10b, 10a becomes part of the data flow.

[00132] Reference is now made to Figure 15, a flowchart of the steps removing a target node 10 for a data flow node 10. As such, Figure 15 is an expanded view of step 265 of Figure 11. Just as nodes 10 can be added as target nodes 10 for data flow nodes 10, thus becoming themselves part of the data flow and adding additional paths to the data flow, it is also possible to remove a given node 10 as a target node 10 from the list of target nodes 10 maintained by a data flow node 10. This may be desirable where a data flow node 10 has not used the target node 10 to transmit EUM data messages or EEC messages to the terminating queue 40 for a period of time exceeding a data flow timeout value. As such, if the target node 10 is removed from the list of target nodes for that data flow node 10, it can be more readily used as part of another EEC. The data flow timeout value is used to ensure that target nodes 10 in the data flow are not left idle for too long. This data flow timeout value should be relatively long compared to the amount of time required to create the EEC from initiating queue 45 on the initiating node 10 to the terminating queue 40 on the terminating node 10 in the first place. Specifically, the data flow timeout period should be at least an order of magnitude greater then the total time needed to establish the EEC initially. The data flow timeout period could also be dynamically adjusted over time based on how much time elapsed between when a target node 10 is removed from the data flow until when it is re-added as a target node 10 for a data flow node 10, thus returning it to the data flow.

[00133] Commencing at step 410, the NCM 20 on the data flow node 10 verifies that the data flow timeout value for the target node 10 has been

exceeded. If not, proceeding to step 415, the target node 10 remains on the list of target nodes 10 for the data flow node 10 and the target node 10 continues itself to be a data flow node 10 for the terminating queue 40. Otherwise, moving to step 420, the NCM 20 sends the target node 10 a message that it is no longer a target node 10 for the data flow node 10 and the NCM 20 removes the target node 10 from it's list of target nodes 10 for the terminating queue 40.

[00134] Proceeding to step 425, the NCM 20 on the former target node 10 will remove the data flow node 10 that sent the message from a list maintained by that NCM 20 of nodes 10 for which the former target node 10 is on the data flow for the terminating queue 40. The NCM 20 on the former target node 10 will also send a message to all target nodes 10 of the former target node 10 to inform them that they are no longer in the data flow as far as the former target node 10 is concerned. However, since the former target node 10 may still be a target node 10 for a different data flow node 10, the NCM 20 of the former target node 10 will only set the former target node's 10 data flow flag to false if it is no longer a target node 10 for any data flow node 10 for the relevant terminating queue 40. This process repeats itself for each target node 10 of the former target node 10 and their successive target nodes 10. The end result is that all the paths commencing from the data flow node 10 that removed the former target node 10 and through the former target node 10 are removed from the data flow.

[00135] Figure 16 is a flowchart of the steps for removing a node 10 completely from all paths on the data flow for a terminating queue 40. In this case, the node 10, other than the initiating node 10, is no longer a target node 10 for any data flow nodes 10, and thus the node 10 ceases to be a data flow node 10 itself. Beginning at step 430, the node 10 determines that it is no longer part of the data flow for the EEC to the terminating queue 40. This may occur in any of the following circumstances:

(a) 1. All other data flow nodes 10 that have designated the node 10 as a target node 10 are disconnected.

(b) Every other node 10 that told that particular node 10 that it is in the data flow for a terminating queue 40 have sent that particular node 10 a message indicating that particular node 10 is no longer in the data flow for the terminating queue 40.

(c) All data flow nodes 10 that told a particular node 10 that it is in the data flow tell that particular node 10 that it is no longer a target node 10.

[00136] Proceeding to step 435, once the NCM 20 on a node 10 determines that the node 10 is no longer part of the data flow to the terminating queue 40 for any data flow nodes 10, the NCM 20 sets that node's 10 data flow flag for the terminating queue 40 to false and transmits a system message to all target nodes 10 that they are no longer in the data flow for the terminating queue 40 from that node 10. Finally, at step 440, the NCM 20 on the node 10 removes all target nodes 10, with exception of the target node 10 having the lowest latency, from it's list of target nodes 10 for that terminating queue 40. This leaves the node 10, now completely removed from the data flow for the terminating queue 40, with only one target node 10 for it's potential route to the terminating queue 40 on the terminating node 10.

[00137] Reference is now made to Figure 17, a flowchart of the steps for optimizing potential routes to a queue of any type for a node 10 outside of the data flow for an EEC. As such, Figure 17 is an expanded view of step 270 of Figure 11. This procedure is carried out by the NCM 20 on the node 10 attempting to optimize it's potential route to the origin 10 node for the queue. While the NCMs 20 of nodes 10 that are not in the data flow do not maintain multiple target nodes 10 for the terminating queue, they do, however, always seek to maintain the lowest latency target node 10, and therefore potential route to the terminating node 10 for the terminating queue 40. This ensures, for EECs, that these nodes 10 can be chosen by data flow nodes 10 to be added to the data flow as required to provide lower latency paths for data flow or when a data flow node 10 is at capacity. It also ensures the lowest node latency potential route for EEC messages for information queues 35 and acknowledgement messages for initiating queues 45, which facilitates verification of reception of EUM data messages and establishment of EECs.

[00138] Beginning at step 445, when a node 10 outside the data flow receives a latency update, it's NCM 20 will examine the node latency and capacity information for the neighbor node 10 that sent the update, as always. The neighbor node 10 sending the latency is a potential target node 10. Proceeding to step 450, the NCM 20 on the node 10 outside the data flow receiving the latency update will determine whether the potential target node 10 is preferable for replacing the current target node 10 based on the following rules:

(a) The NCM 20 of a node 10 outside the data flow will always prefer a node 10 not at capacity over a node 10 at capacity, regardless of node latency.

(b) If both nodes 10 under examination, i.e. the current target node 10 and a potential target node 10, are at capacity or not at capacity, the node 10 outside the data flow will prefer the lowest node latency node 10 of the two nodes 10 under examination.

[00139] Moving to step 455, if the potential target node 10 is not preferable to the current target node 10, the NCM 20 on the node 10 receiving the latency update maintains the current target node 10 for the queue. Otherwise, advancing to 460, the NCM 20 on the node 10 outside the data flow will attempt to determine whether the potential target node 10 would create a loop. In this case, however, loop test GUIDs are not used. Constantly testing potential target nodes 10 for loops using loop test GUIDs on all nodes 10 outside the data flow would quickly cause the network 5 to be overrun with loop test GUID messages. Instead, a latency loop test that compares the node latencies of the potential target node 10 and the current target node 10 over time is used.

[00140] In the circumstance where a potential target node 10, if actually chosen as the new target node 10, would create a loop, the major cause of apparent lower node latency is lag introduced by the travel time for data in the loop between the node 10 outside the data flow and the potential target node 10. In other words, depending on the number of nodes 10 in a loop and the

respective node latencies of each node 10 in the loop, it is possible that latency updates which would indicate that the potential target node 10 in fact has a higher node latency than the current target node 10 would not yet have had time to propagate this higher node latency value to the node 10 outside the data flow considering the potential target node 10. For example, if every second the node latency of the node 10 outside the data flow increases by 1 second, and there is a potential loop with a three second lag between the node 10 outside the data flow and the potential target node 10, the potential target node 10 would appear to have a latency at least three seconds lower than the current target node 10 at any given moment. Thus, it is important to monitor the latencies not at one given moment, but over a period of time.

[00141] The NCM 20 on the node outside the data flow conducts the latency loop test by comparing the maximum latency of the current target node 10 and the maximum latency of the potential target node 10 over two different, yet overlapping time periods. More specifically, the potential target node's 10 time period would commence before, and finish an equal amount of seconds after, the current target node's time period. In other words, if the current target node's 10 time period was CTNTP seconds long and M is the length of the time period both immediately preceding and following CTNTP, the potential target node's 10 time period, or PTNTP, would be:

$$PTNTP = CTNTP + 2M$$

[00142] Should the maximum node latency recorded during PTNTP the for a potential target node 10 ever equal or exceed the maximum node latency for current target node 10 for the queue during CTNTP, the NCM 20 on the node 10 outside the data flow will conclude there is a loop. If so, returning to step 455, the NCM 20 on the node 10 outside the data flow will retain the current target node 10. Otherwise, advancing to step 465, the NCM 20 on the node 10 outside the data flow will remove the current target node 10 from the list of target nodes 10 for the queue and instead substitute the potential target node 10 as the sole target node 10 for the queue.

[00143] The size of the time periods for the latency loop test is based on the smallest time period with the minimum number of observations to be effective. In other words, while it is not critical if the time periods for the latency loop test are too long, a loop may be missed if the time period is too short. For example, one could arbitrarily choose a minimum of 5 observations over a minimum CTNTP of five seconds. However, the actual time periods chosen should reflect network 5 size and traffic. If a network 5 is supporting a very large number of queues, it will take a relatively large amount of time for node latency information to propagate through nodes 10 outside the data flow because nodes 10 pass these latency updates in a bandwidth throttled manner. In this circumstance the CTNTP would need to be expanded.

[00144] Figure 18 is a flow chart of the steps for resolving loops created in nodes 10 outside the data flow. Beginning at step 470, a loop is created by nodes 10 outside the data flow. Proceeding to step 475, the loop will cause the node latency of all the nodes 10 in the loop to spiral upwards. In such a situation, the NCM 10 on the node 10 with the highest node latency will propagate that value, via latency updates, to it's target node 10 for the queue 40. The NCM 20 on that target node 10 will use this value to calculate it's own node latency, thus further increasing the value and forward it on to it's target node 10 for it's next latency update. This process continues and causes the upward spiral in node latency for nodes 10 in the loop. Next, at step 480 each node 10 in the loop will attempt to identify lower latency nodes 10 as target nodes. To identify the target nodes 10, the steps for optimizing potential routes, shown in Figure 18 as an expanded view of step 270 of Figure 11, are executed. As the latency of each node 10 is increasing rapidly, this will facilitate identification of lower node latency options for target nodes 10. Proceeding to step 485, once lower node latency target nodes 10 are identified, they can replace the target nodes 10 that cause the loop and the loop can be resolved.

[00145] Loops in nodes 10 outside the data flow will nonetheless be rare due to the use of the use of the latency loop test, shown as step 460 of Figure

17, when choosing new target nodes 10. Further, since explicit probing of potential target nodes 10 and their paths is conducted using the loop GUID test, shown in Figure 13, loops will also be created rarely created in the data flow. The only occurrence of a loop in the data flow that might occur would be due to intervening changes within a path leading from a target node 10 chosen by a data flow node 10 after the loop GUID test has been executed. Whether in the data flow or not, any loops accidentally created will be resolved quickly. Any such loops will be resolved by either having a node 10 within the loop pick a new target node 10 that does not loop or by having a target node 10 that commences the loop be removed as a target node 10 by all neighbor nodes 10 that had previously designated it as a target node 10, thus removing access to the loop entirely. Since the node latency of any nodes 10 that are part of a loop will either spiral quickly upwards or be set to infinity, NCMs 20 on nodes 10 will quickly choose other neighbor nodes 10 as target nodes 10, thus ensuring that very few EUM messages will ever be sent down a loop.

[00146] It will be apparent to one skilled in the art that adding additional target nodes 10 to the data flow, optimizing potential routes, and removing unused target nodes provide an adaptive capacity for the present invention in terms of use of bandwidth and efficiency for transmission of EUM messages. Addition of target nodes 10 for data flow nodes 10 creates additional paths for transmitting messages in the data flow, especially when a data flow node 10 is at capacity. Optimization of potential routes ensures that a new target node 10 added to the data flow should provide optimal node latency and that potential routes to information queues 35 and initiating queues 45 provide the lowest node latency. Removal of unused target nodes 10 from the data flow ensures that unused resources and network bandwidth can be made available for other connections, once again on an adaptive basis. As always, this adaptive capacity and efficiency is achieved without maintenance of any global knowledge of the topology of the entire network 5. At the same time, the avoidance of loops further enhances efficiency of transmission. Furthermore, the constant maintenance of potential routes and at least one

path of target nodes 10 for the data flow ensure that reliability is not compromised. Once again, these benefits are achieved without maintenance of any global knowledge of the topology of the entire network 5.

[00147] Reference is now made to Figure 19, a flowchart of the steps for determining when a node 10 may send messages to another node 10. As such, Figure 19 is an expanded view of step 60 at Figure 4. While the data flow and potential routes establish where a node 10 may send EUM messages, they do not determine when each node 10 can send to another node 10. The determination of when and whether a node 10 will send messages to another node 10 is effected in two steps. At step 490, a flow control measures using tokens assure that the receiving node 10 for a transmission has enough capacity to receive and store the messages. At step 495, for data flow node's in an EEC only, a latency comparison simultaneously ensures that a sending data flow node 10 will only send EUM messages to a given target node 10, and therefore itself a data flow node 10, if the given target node 10 provides a node latency for the terminating queue 40 that is useful. Whether the given target node's 10 node latency is useful is determined by evaluating the node latency of other target nodes 10 for the sending data flow node 10 and the queue latency (QLatency) of the terminating queue 40 on the sending data flow node.

[00148] Figure 20 is a flowchart of the steps for flow control. As such, Figure 19 is an expanded view of step 490 of Figure 19 for flow control. Each node 10 has a variable amount of memory, primarily RAM, used to store information relevant for sending and receiving messages to and from other nodes 10 and queues. Thus, it is important to control when, and how much, information a node 10 can send to a neighbor node 10 at any given moment.

[00149] In the present invention, flow control operates using the mechanism of tokens. The NCM 20 on a receiving neighbor node 10 will transmit to a sending neighbor node 10 a number representing an amount of tokens corresponding to the number of bytes that the sending neighbor node 10 can send to the receiving neighbor node 10. The sending neighbor node

10 is not allowed to send more bytes than this number. When the receiving neighbor node 10 has more resources available and it is informed by the NCM on the sending neighbor node 10 that the sending neighbor node 10 is getting low on tokens, the receiving neighbor node 10 can transmit more tokens to the sending neighbor node 10. Tokens are allocated and accounted for using token updates, which also contain latency updates. All quota calculations and generation of token updates and flow control related messages are handled by the NCM 20 on each node involved.

[00150] There are two levels of flow control. The first is node-to-node flow control and the second is queue-to-queue flow control. Node-to-node flow control is used to constrain the total number of bytes of data for any kind of messages that is sent from the sending neighbor node 10 to the receiving neighbor node 10. Node-to node flow control applies to all nodes 10 and both EUM messages and system messages. Queue-to-queue flow control is used to constrain the number of bytes of EUM messages that move from a queue in the sending neighbor node 10 to a queue in the receiving neighbor node 10 having the same queue label. Thus, a sending neighbor node 10 can send 10 bytes of messages for a given queue to a receiving neighbor node 10 only if the sending neighbor node has 10 ten tokens for node-to-node flow control to that receiving neighbor node and 10 tokens for queue-to-queue flow control for the given queue on the receiving neighbor node 10. This transmission will also cost the sending neighbor node 10 ten tokens in the node-to-node flow control as well as 10 tokens in the queue-to-queue flow control for that particular queue for that particular receiving neighbor node 10. In other words, to send a given queue on the receiving neighbor node 10 a message of X bytes, the amount of tokens available on the sending neighbor node 10 for the given queue on the receiving neighbor node 10 for queue-to-queue flow control and the amount of tokens available on the sending neighbor node 10 for node-to-node flow control must both be at least equal to X.

[00151] To manage tokens, the NCMs 20 on the sending neighbor node 10 and receiving neighbor node 10 use a variety of variables. The maximum

amount of tokens that the receiving neighbor node 10 may allocate to the sending neighbor node 10 is known as the outstanding tokens limit (OTL). The maximum amount of bytes that the sending neighbor node 10 may send at any one time is referred to as the quota and represents the amount of outstanding tokens that have not been used by the sending neighbor node 10. For every byte sent, the sending neighbor node 10 subtracts one token from the quota. In other words, if the quota is stored in a variable QuotaSender, when the sending neighbor node 10 sends a message having a size of MessageSent, the quota is calculated as follows:

QuotaSize = QuotaSize – MessageSent

[00152] At the same time, the receiving neighbor node 10 attempts to estimate the amount of outstanding tokens remaining for the sending neighbor node 10. This is the outstanding tokens estimate (OTE) and is calculated on the receiving neighbor node 10 by subtracting the size of messages, in bytes, received by the receiving neighbor node 10 from the amount of tokens allocated. The size of the messages received may be stored in a variable MessagesReceived. Since the amount tokens allocated is initially the OTL, the OTE is initially set at the value of OTL. Thus, when the receiving neighbor node 10 receives a message from the sending neighbor node, it's NCM 20 calculates OTE as follows.

OTE = OTE - MessageReceived

[00153] The NCM 20 on the receiving neighbor node 10 also maintains an OTL version number and a quota request flag. The OTL version is initially set at 0 when the sending neighbor node 10 and the receiving neighbor node 10 first commence negotiating transmission of messages between the sending node 10 and receiving node 10. It is then incremented by one every time the OTL is changed. The quota request flag is initially set to false and is set to true every time the receiving node 10 receives an additional quota request from the sending neighbor node 10. The quota request flag is set to false when the additional quota request has been processed. The NCM 20 on

the sending neighbor node 10 also maintains a copy of the current OTL version number as well as a copy of the last quota request version number. Whenever the sending neighbor node 10 receives a quota update from the receiving neighbor node 10, the NCM 20 on the sending node 10 changes the current OTL to the OTL version sent by the receiving neighbor node 10. At the same time, whenever the sending neighbor node 10 requests additional quota it increments the last quota request number to the current quota request number. In this fashion, the sending neighbor node 10 knows it has sent an additional quota request and will not constantly repeat the same request while waiting for the receiving neighbor node 10 to process the additional quota request.

[00154] A detailed explanation of the steps for flow control is now provided. To facilitate comprehension of the process, a number of examples of values for the variables on both the sending node and receiving node are furnished in Tables 15 - 21.

[00155] Commencing at step 500, the sending neighbor node 10 and receiving neighbor node 10 commence negotiation, via their NCMs 20, so that the sending neighbor node 10 may transmit messages to the receiving neighbor node. As shown in Table 15, all of the numerical values for sending and receiving node are at 0 and all flags are set to false.

Table 15

Sending neighbor node		Receiving neighbor node	
OTL Version	0	OTL Version	0
Previous OTL version	0	OTL	0
Quota (QuotaSize)	0	OTE	0
		Quota request flag	False

Proceeding to step 505, the NCM 20 on the receiving neighbor node 10 allocates a small, default number of tokens to the sending node 10 and

increments the OTL version to 1. It then sends a token update with the amount of tokens allocated and the current OTL version to the sending neighbor node 10. For the purposes of example, and as shown in Table 16, the default number of tokens is set at 1. Since the sending node 10 has not yet used the token, the quota on the sending node 10 after the token update is also 1, as are the values for the OTL and the OTE.

Table 16

Sending neighbor node		Receiving neighbor node	
OTL Version	1	OTL Version	1
Previous OTL version	0	OTL	1
Quota (QuotaSize)	1	OTE	1
		Quota request flag	False

[00156] Proceeding to step 510, once the sending neighbor node 10 has received some tokens for the quota value, the sending neighbor node 10 may now send a message to the receiving neighbor node 10 provided that the size of the message to be sent, which may be stored in a variable MessageSize, is less than the amount of the quota value. In other words, at this step, the NCM on the sending neighbor node 10 verifies whether the following is true:

Quota >= MessageSize

[00157] Turning to step 515, if the quota is not greater than or equal to the size of the message to be sent, then the NCM 20 on the sending neighbor node 10 will send an additional quota request to the receiving neighbor node 10. Proceeding to step 520, when the receiving neighbor node 10 receives the request additional quota request, the receiving neighbor node's 10 NCM sets the quota request flag to true. The NCM 20 on the receiving neighbor node 10 also verifies whether it has encountered a no data to send condition, wherein another neighbor node 10 of the receiving neighbor node 10 was

ready to receive data transmitted by the receiving neighbor node 10, but the receiving neighbor node 10 had no data to send.

[00158] Continuing to step 525, if the NCM 20 on the receiving neighbor node 10 has encountered a no data to send condition it will decide the amount of tokens by which it will increase the OTL and increases the OTL and the OTE for the sending neighbor node 10 by this amount. Since the OTL is increased, the OTL version is incremented by one. Once the receiving neighbor node 10 increases the OTL, it resets the quota request flag to false. The receiving node 10 then sends a message, as a token update, containing the amount of tokens by which the OTL was increased and the new OTL version to the sending neighbor node 10.

[00159] Moving to step 530, when the sending neighbor node 10 receives the token update, the NCM 20 on the sending neighbor node 10 increments the quota by the amount by which the OTL was increased. It also copies the new value, incremented by one, for the OTL version. Turning to step 535, if the NCM 20 on the receiving neighbor node 10 has not encountered a no data to send condition, then the receiving neighbor node 10 will wait until such a condition is generated before increasing the OTL.

[00160] For the purposes of example, suppose that, as shown in Table 16, the quota value on the sending neighbor node 10 is 1 and the sending neighbor node 10 has a message of five bytes to send. Suppose further that the receiving neighbor node 10 has encountered a no data to send condition, and that the receiving neighbor node's 10 NCM 20 has been programmed to increase the OTL value by 100, if possible, for each additional quota request. When the sending neighbor node 10 tests to see whether the quota size is sufficient at step 510, the result will be false. Thus, the sending neighbor node 10 will send an additional quota request to the receiving neighbor node 10 at step 515 and increment the previous OTL value of 0 to the match the OTL value of 1. When the receiving neighbor node 10 receives the request it sets the quota request flag to true at step 520. Thus, for the purposes of the

example, the values of the variables on the sending neighbor node 10 and receiving neighbor node 10 at step 520 are shown in Table 17.

Table 17

Sending neighbor node		Receiving neighbor node	
OTL Version	1	OTL Version	1
Previous OTL version	1	OTL	1
Quota (QuotaSize)	1	OTE	1
		Quota request flag	True

[00161] Since the NCM 20 on the receiving neighbor node 10 has encountered a no data to send condition, at step 520, it proceeds to step 625, and increases the value of OTL by 100. The receiving neighbor node 10 also increments the OTE by the same amount and increments the OTL Version value by 1. The NCM 20 on the receiving neighbor node 10 then sets the quota request flag to false. It then sends a token update to the sending neighbor node 10.

[00162] At 530, the sending neighbor node 10 receives the token update and it's NCM 20 increments the sending neighbor node's 10 copy of the OTL version to the value sent in the token update, i.e. the same OTL value stored on the receiving neighbor node 10. The NCM 20 on the sending neighbor node 10 also increases it's quota by the amount indicated in the update, in other words, the amount by which the OTL was increased on the receiving neighbor node 10. The values of the variables on the sending and receiving neighbor node 10 after step 530 are indicated in Table 18.

Table 18

Sending neighbor node		Receiving neighbor node	
OTL Version	2	OTL Version	2

Previous OTL version	1	OTL	101
Quota (QuotaSize)	101	OTE	101
		Quota request flag	False

[00163] Moving to step 540, if at any point the sending neighbor node's 10 NCM 20 determines that the quota is greater than or equal to the size of the message to be sent, the sending neighbor node 10 sends the message and the quota is decremented by the size of the message sent. Similarly, when the receiving neighbor node 10 receives the message, it's NCM 20 reduces the size of the OTE by the size of the message received. Thus, for the purposes of example, if the sending node had a message of five bytes to send and the variables of the sending neighbor node 10 and receiving neighbor node 10 had the values set out in Table 18, the sending node would be able to send the message. After the message was received by the receiving neighbor node 10, the variables on the sending neighbor node 10 and receiving neighbor node 10 would have the values set out in Table 19.

Table 19

Sending neighbor node		Receiving neighbor node	
OTL Version	2	OTL Version	2
Previous OTL version	1	OTL	101
Quota (QuotaSize)	96	OTE	96
		Quota request flag	False

[00164] Proceeding to step 545, as mentioned earlier, when the receiving neighbor node 10 receives a message from the sending neighbor node 10 it attempts to refresh the quota to ensure that the number of tokens allotted is replenished as messages are received. This is useful for

attempting to ensure that the sending neighbor node 10 has a constant supply of tokens, up to the OTL value.

[00165] To refresh the quota, the receiving neighbor node 10 verifies, whenever a message is received from the sending neighbor node 10 and the size of the OTE has been decremented, whether the OTE is less than 50 per cent of the OTL and the receiving neighbor node 10 is capable of receiving more messages. Should this be the case, the receiving neighbor node 10 automatically will send a token update to allot more tokens to the sending neighbor node 10. At the same time, the receiving neighbor node 10 will increment the OTE value by the amount by which it increased the quota. Should the OTE be 50 percent or more of the OTL, then the receiving neighbor node 10 may allot more tokens on a discretionary basis.

[00166] The actual amount of tokens allocated may vary, depending on the amount of data that the receiving neighbor node 10 is capable of receiving, both in it's queue and it's physical memory. However, in all cases, the receiving neighbor node 10 will not allow the sum of the OTE and the number of new tokens allocated to exceed the OTL. In other words, the maximum new quota allotment (MNQA) at any one time is calculated as follows:

$$MNQA = OTL - OTE.$$

[00167] Further, if the receiving neighbor node 10 is a data flow node 10 and is at capacity, it will attempt to give all sending neighbor nodes 10 that are also data flow nodes 10 for the terminating queue 40 enough tokens to ensure that the receiving data flow node 10 will be kept at capacity if possible. However each sending data flow node 10 to that receiving node 10 must not get more than their maximum OTL once the receiving data flow node 10 has reached capacity. Otherwise, quota will needlessly be allocated for sending to the receiving data flow node 10 when the receiving data flow node 10 is already at capacity.

[00168] To surmount this problem, an over capacity token count variable (OCTC) is maintained by the NCM 20 on the receiving data flow node 10 for

all sending data flow nodes 10 for the terminating queue 40. For each byte received from the sending data flow node 10 while the receiving data flow node 10 is at capacity, the OCTC for that terminating queue 40 for that particular sending data flow node 10 is incremented by one. If the receiving data flow node 10 ever stops being at capacity, then the OCTC for all sending data flow nodes 10 is reset to 0.

[00169] When allocating quota during the quota refresh process, the NCM 20 on the receiving data flow node 10 will subtract the OCTC from the OTL to determine the amount by which to increase the tokens for the quota and the OTE. To ensure that tokens are as well distributed is possible, however, whenever data is removed from the terminating queue 40 on a receiving data flow node 10, the number of bytes removed is subtracted as evenly as possible from all OCTCs greater than zero.

[00170] For the purposes of example, suppose that 120 bytes are removed from the terminating queue 40 on a receiving data flow node 10 and there are four sending data flow nodes 10. The OCTC values for each of the sending neighbor nodes 10 are as follows: 0, 100, 20, 50. First, the number of bytes removed is divided by the number of OCTC variables that have a value greater than 0, namely 3. This yields a subtraction figure of 40. If there is an OCTC above 0 and less than the subtraction figure, then we subtract the lowest OCTC value that is less than the subtraction figure from all OCTC values greater than 0. In the example, since the lowest OCTC value, namely 20, is less than the subtraction figure of 40, we subtract 20 from all of the OCTC values that are above 0. This yields the following OCTC values: 0, 80, 30, 0. If the total amount subtracted from all OCTC figures is less than the amount removed from the terminating queue 40, then the process is repeated. In the example, only 60 bytes of the 120 bytes have been subtracted. Thus, the process is repeated for the remaining 60 bytes. As there are now two OCTC variables with a value above 0, the new subtraction value is 30. When this is applied to the OCTC values, it yields the following OCTC values: 0,30,0,0. As there are no bytes left over, the process of distributing the bytes

among the OCTC values is finished. The resulting OCTC values are then used by the NCM 20 to calculate the amount by which the quota may be increased for each sending data flow node 10, as well as the OTE for the receiving data flow node 10. Thus, for data flow nodes 10, the MQNA for each sending data flow node 10 to the receiving data flow node 10 will be calculated as follows:

MNQA = OLT – OCTC – OTE.

[00171] For all nodes 10 and messages of all types, the process of refreshing quota by the receiving neighbor node 10 is continuous. The receiving neighbor node's 10 NCM 20 will always attempt to allocate new tokens to refresh the amount of outstanding tokens unless the OTL is reached.

[00172] For purposes of example, suppose that the variables on the sending neighbor node 10 and receiving neighbor node 10 node have the values set out in Table 19. Suppose further that the sending neighbor node 10 sends a message of 52 bytes, that the receiving neighbor node 10 has received the message, and that the receiving neighbor node has been programmed to attempt to refresh the amount of tokens by MNQA. Thus, after step 625, the new values of the variables would be as set out in Table 20.

Table 20

Sending neighbor node		Receiving neighbor node	
OTL Version	2	OTL Version	2
Previous OTL version	1	OTL	101
Quota	49	OTE	49
		Quota request flag	False

[00173] Since the value of the OTE is less than 50% of the OTL, then the receiving node must act to refresh the quota. Since the receiving

neighbor node 10 has been programmed to try to increment the quota by MNQA, it's NCM 20 will thus send a quota update refreshing quota in the sending node to match the OTL. Thus, after the quota is refreshed at step 630, the variables on the sending neighbor node 10 and receiving neighbor node 10 will be as set out in Table 21.

Table 15

Sending neighbor node		Receiving neighbor node	
OTL Version	2	OTL Version	2
Previous OTL version	1	OTL	101
Quota	101	OTE	101
		Quota request flag	False

[00174] The steps for flow control shown in Figure 20 apply to both node-to-node flow control and queue-to-queue flow control. Further, the variables listed above for flow control are maintained on the sending and receiving node for both all messages between nodes 10, i.e. node-to-node flow control, as well as for each queue to which the sending neighbor node 10 sends data on the receiving neighbor node 10, i.e. for queue-to-queue flow control. The variables are maintained and adjusted independently one from the other.

[00175] The OTL is also constantly shrunk by the NCM 20 on the receiving neighbor node 10 at a small but fixed rate. For example, it could be shrunk at a rate of one percent per second. This allows automatic correction over time for OTLs that may have grown too high. In any event, if this constant shrinking drops the OTL too low, the NCM 20 on the sending neighbor node 10 can always request additional quota, as shown at 600, to ensure that the OTL is increased.

[00176] Token updates for different queues can be grouped together. Thus if one token update needs to be sent, token updates for other queues 35 on the receiving neighbor node 10 can instantly be generated for a small amount of tokens can be sent at the same time. This will reduce the incidence of a special message being sent with just one quota update.

[00177] As will be apparent to one skilled in the art, flow control facilitates transmission of messages by ensuring that nodes 10 are not

overrun with data. As such, it increases reliability of the network 5 while also providing a mechanism for controlling which nodes 10 may transmit messages to a given neighbor node 10, as well as for a particular queue on that neighbor node 10 at any given time.

[00178] Figure 21 is a flowchart of the steps for determining which data flow nodes 10 are useful target nodes 10 using a latency comparison. As such, Figure 21 is an expanded view of step 495 of Figure 19. Provided the sending data flow node 10 has sufficient tokens for a receiving data flow node 10 that is a target node 10, the sending data flow node 10 is eligible to send to that receiving data flow node 10. However, even if there are multiple target nodes 10, they may not all necessarily be used for sending of data at any given time. Before sending EUM data messages to a receiving data flow node 10 that is a target node 10, the sending data flow node's 10 NCM 20, will apply a latency comparison to determine which target nodes 10 are most useful for sending given the queue latency (QLatency) of the terminating queue 40 on the sending data flow node 10 and the node latencies of all target nodes 10.

[00179] Commencing at step 550, a sending data flow node 10 has EUM messages to send to a terminating queue 40. Next, at step 555, the NCM 20 on the sending data flow node 10 effects a latency comparison for all target nodes 10 for that terminating queue 40. For each potential target node 10, the sending data flow node 10 initially tests the following relation:

> QLatency on sending data flow node >= NLatency target node being being tested – min([NLatency of all target nodes])

[00180] Proceeding to step 560, if the relation is true, then the sending data flow node 10 will transmit the EUM message to the target node 10, provided the sending data flow node 10 has sufficient tokens. However, for every Y seconds that the target node's 10 node latency is over the minimum of all the target node 10 node latencies, the NCM 20 will ensure that the sending data flow node 10 will store Y seconds of EUM messages in it's copy of the terminating queue 40 before using that target node 10. This ensures

that the data for the EUM message can be sent to the lowest latency target nodes 10 first while the sending data flow node 10 waits Y seconds. Turning to step 565, should the relation not hold true, then this target node 10 will not be used to send EUM messages at the current time and the sending data flow node 10 node will then look at the next potential target node 10.

[00181] Optionally, if a target node 10 has a node latency above the current queue latency for a sending data flow node 10, then the NCM 20 on the sending data flow node 10 may send a message to that target node 10 asking it to inform the sending data flow node 10 when the node latency of that target node 10 drops below a specified value. Asking the target node 10 to send an update at a specified value will also cause the NCM 20 on the target node 10 to send the current node latency. This solves the problem of rapid updates required to keep the sending data flow node 10 informed as to the latency of the target node 10. If no EUM messages are sent to a target node 10 for a certain time period then, as described previously, that target node 10 is removed from the list of target nodes 10 for that data flow node 10.

[00182] Use of the latency comparisons at step 555 in this fashion ensures that, of all the possible target nodes 10 available to a data flow node 10 for transmitting EUM messages, the target node 10 that will most immediately be able to send the EUM messages will be chosen. Further, it ensures that the sending data flow node 10 waits an appropriate amount of time to send data given the node latency of a given target node 10. Thus, the present invention not only ensures that the lowest node latency neighbor nodes 10 are selected as target nodes 10 for sending EUM messages for a terminating queue 40, it also ensures that an appropriate target node 10 is chosen for actually sending these EUM messages at any given moment.

[00183] For a very large network 5, with a large variation in interconnect speed and node capability, step 70 for queue propagation of Figure 5, an expanded view of which is shown in Figure 5, may be altered to facilitate establishments of EECs, even if there are millions of queues. Very large networks provide three particular challenges for queue propagation. First, the

bandwidth required to keep every node 10 informed of all queues grows to a point where there is no bandwidth left for data. Second, if bandwidth throttling on latency updates, used to propagate queues, is used to ensure priority to EUM messages, this may slow the propagation of queue information required for establishing EECs. Third, nodes 10 with insufficient memory to hold every queue may lose knowledge of a queue. If such nodes 10 are the only target node 10 for a queue at a certain point in a potential route or the data flow for an EEC, it will be impossible for some nodes 10 to send EUM messages to that queue and have the EUM messages reach that queue's origin node 10.

[00184] To overcome this problem, it is necessary to determine the nodes 10 that constitute the core, not shown, of the network 5. The core of the network 5 contains the nodes 10 that have the greatest capacity of memory and bandwidth in the network 5, and thus will be best able to maintain and propagate queues. In general, the core of the network 5 is most likely to be centrally located topologically within the network 5.

[00185] Since nodes 10 not at the core of the network 5 will generally not have as much memory or bandwidth as nodes 10 at the core 5, they are the nodes 10 that may be forced to delete knowledge of a queue or which may not receive latency updates quickly enough for a queue to allow them to establish knowledge of a queue which is required for sending EUM messages to that queue and establishing EECs. Conversely, if a node 10 not at the core of the network is an origin node 10 for a queue, there may be significant delays, due to low capacity, in propagating knowledge of the queue to the nodes 10 in the core. Thus, it is necessary to locate the core, where the nodes 10 are most likely to be able to maintain queues and propagate latency updates and token updates for queues the most quickly, and to establish a priority path to the core for nodes 10 that are not at the core.

[00186] The present invention does not maintain any global network topology to locate the core. Thus, steps toward the core of the network 10 are identified by the NCM 20 by consulting each neighbor node 10 for each node 10 to determine which neighbor node 10 is a target node 10 for the most

queues. A neighbor node 10 is picked as a target node 10 because it has the lowest node latency for a queue, and the lowest node latency will generally be provided by the fastest node 10 that is closest to a terminating node 10 for a queue. If a given neighbor node 10 for a given node 10 is used as a target node 10 for more queues then any other neighbor node 10 for a given node 10, then the given neighbor node 10 probably is higher capacity than the other neighbor nodes 10 and represents a potential step toward the core of the network 5.

[00187] The priority path that is established to the core is referred to as a high speed propagation path (HSPP), not shown, for a queue or class of queues and is used to rapidly propagate a queue from it's origin node 10 to and from the nodes 10 at the core of the network. If a node 10 is in a HSPP for a particular queue it's NCM 20 will immediately process and send any of the following three types of messages:

(a) If the queue is a new queue, any latency update to propagate knowledge of the queue;

(b) Latency updates to indicate that the node latency for the queue has become infinity; and

(c) Latency updates when node latency for the queue has moved from infinity to some other value.

[00188] The NCM 20 on the node 10 sends these messages immediately to all it's neighbor nodes 10, or at a minimum those nodes directly in the HSPP. This ensures that all nodes 10 in the HSPP will always know about the queue in question, provided they have received at least one latency update informing them of the queue and have sufficient memory capacity to maintain their own version of the queue. The HSPP does not in itself contain knowledge of the queues. Rather, the HSPP only sets up a path with a very high priority for transmission of the three updates mentioned above.

[00189] The HSPP path is bi-directional. When a queue, of any kind, is first created on an origin node 35, the step 70 for queue propagation of Figure

5, an expanded view of which is shown in Figure 5, will first be used. This will give any nodes 10 that are in close proximity to the origin node 10 a chance to get the most direct potential route or path for data flow to the origin node 25 for that queue. After a certain amount of time has elapsed (20 seconds for example) the NCM 20 on the origin node 10 will then set up an HSPP for the queue to the core of the network 5 to facilitate queue propagation. In the particular case of EECs, an NCM 20 on the initiating node 10 will also create an HSPP if it has received an message in it's initiating queue 45 informing it of the queue label of the terminating queue 40, but has not yet received a latency update for the terminating queue 40 for queue propagation purposes. As soon as the EEC is established, the NCM 20 on the initiating node 10 will remove the HSPP. An HSPP created by an initiating node 10 will only extend to the first node 10 encountered to which the terminating queue 40 has been propagated. If the queue on an origin node 10 for which an HSPP is established is removed on that origin node 10, the NCM on the origin 10 will also remove it's HSPP into the core for the queue. Otherwise it will maintain the HSPP.

[00190] HSPPs are established by using HSPP markers. If a node 10 is told of an HSPP, by receiving an HSPP marker, it's NCM 20 must remember that HSPP until it is told to forget that HSPP, the neighbor node 10 that told it of the HSPP becomes inaccessible, or the connection for which an HSPP is established is terminated. The NCM 20 on each node 10 will set aside a certain amount of memory for HSPP markers. The NCM 20 on each node 10 will then provide tokens to all neighbor nodes 10 that allow those neighbor nodes 10 to send this node 10 an HSPP marker. These tokens are referred to as HSPP tokens. If a node 10 stores an HSPP marker, it's NCM 20 must also reserve space to store information associated with the queue associated with the HSPP, and some space with which to move messages associated with that queue.

[00191] The same actions undertaken for flow control in step 490 of Figure 19, an expanded view of which is provided in Figure 20, are used to

regulate the amount of HSPP tokens given to a particular neighbor node 10. If the NCM 20 on a node 10 has asked for more HSPP tokens and has not received them after a small fixed time, the node 10 whose NCM 20 refused to send more tokens will be marked as full. The NCM 20 on a node 10 will pick the neighbor node 10 that is not full and that has the most target nodes 10 associated with it as the recipient for an HSPP marker, thus bringing the neighbor node 10 into the HSPP. This neighbor node 10 will most likely to point toward the core of the network 5. The NCM 20 on the chosen neighbor node 10 will then do the same for it's neighbor nodes 10 to establish the next step in the HSPP. Since HSPPs are bi-directional, the NCM 20 on a node receiving an HHSP marker will mark the neighbor node 10 that sent the HHSP marker as a source node 10 for the HHSP and the chosen neighbor node to which the node 10 sends an HHSP marker as a destination node 10 for the HHSP. When choosing a neighbor node 10 as a destination node 10 for the HSPP, the node 10 will first execute tests for instant loops and GUID loop test procedures similar to those used in step 255 of Figure 11, an expanded view of which is shown in Figure 12.

[00192] The basic rules for choosing a neighbor node 10 as a destination node 10 to inform about an HHSP are now presented. As mentioned, the NCM 20 on a node 10 will attempt to choose the neighbor node 10 that is not full, does not create a loop, and has the most target nodes 10 associated with it as the destination node 10 for an HSPP marker, thus bringing the neighbor node 10 into the HSPP. If a node 10 receives an HHSP marker and can transmit an HHSP marker to the destination node 10, the NCM 20 on that node 10 will do so as soon as possible. However, the NCM 20 on a node 10 will only choose one neighbor node 10 as a source node 10 for receiving data over an HHSP no matter how many nodes 10 tell it about the HSPP by sending it an HHSP marker. In addition, the NCM 20 on a given node 10 will only send an HHSP marker to a given neighbor node 10 that it considers to be the destination node 10 and the given neighbor node 10 can not be the same neighbor node 10 whose NCM 20 that sent the given node 10 it's HHSP marker. Should a given neighbor node 10 that the NCM 20 on

a given node 10 considers the destination node 10 be the same neighbor node 10 whose NCM 20 first sent the given node 10 an HSPP marker for the HSPP, the node 10 will simply not send an HSPP marker to any nodes 10. This helps avoid the creation of loops.

[00193] The NCMs 20 on nodes 10 track the status of HSPPs on the basis of whether they are transmitting messages, i.e. are active, or whether they are not transmitting data, i.e. they are not active. If an HHSP on a node 10 is active, then a flag in the HHSP marker on the node 10 will be set to true, otherwise it is false. If the NCM 20 on a given node 10 has sent an HHSP marker for a queue to a destination node 10 and the NCM 20 on the given node 10 subsequently tells the destination node 10 that the HSPP is not active, then the NCM 20 on the destination node 10 will also tell it's own destination node that the HSPP is now not active. However, if an HSPP on a node 10 becomes not active or the node 10 is informed that an HSPP has become not active, and that node's NCM 10 has not yet sent an HHSP marker to it's destination node 10, it will abstain from sending an HHSP marker to it's destination node 10. Thus, the destination node 10 will not be informed of the HSPP.

[00194] If an HHSP on a given node 10 goes from active to not active, or the connection is lost to the given node's source node 10, the given node's 10 NCM will tell it's destination node 10 that the HHSP is now not active 10. Next, the NCM 20 on the given node 10 will wait a pre - determined amount of time, for example 20 seconds. Next, if the HSPP has become active again on the source node 10, then the NCM 20 on the given node 10 will change the it's status for the HSPP to active and will inform the destination node 10 that the HSPP is now active again on the given node 10. However, if the HHSP is still not active on the source node 10, but the NCM 20 on the given node 10 can identify a neighbor node 10 on which the HHSP 10 is still active, then the NCM 20 on the given node 10 will choose that neighbor node 10 as a new source for HSPP messages for that HHSP. If however, there is no neighbor

node 10 on which the HSPP is still active, then the NCM 20 on the given node 10 will delete the HSPP marker and will lose knowledge of the HSPP.

[00195] In addition to HSPPs, it may be desirable that bandwidth throttling be more comprehensively used in a larger network 5. In general, total system message bandwidth is limited to a percent of the maximum bandwidth available for all messages of all types. However, high priority and medium high priority system messages will not be limited by the NCM 20 as a percent of maximum bandwidth. For high priority system messages, EUM messages will be ignored and these messages will be sent immediately. Medium high priority system messages will be embedded in a group of EUM messages by the NCM 20, with this group of EUM messages being sent before other groups of messages. All bandwidth throttling is carried out by the NCM 20 on each node 10.

[00196] The remaining messages are classified as low priority and are bandwidth throttled in that they will only be allowed to occupy a percentage of maximum bandwidth at any time. For example, one could specify 5% of maximum bandwidth with a minimum size of 4 Kilobytes for a message. If the total bandwidth available is 10MB/s, this would mean that a 4 Kilobyte packet of system messages would be sent every 0.0819 seconds since 4096 bytes/ (10MB/s * 0.05) = 0.0819 seconds. Thus, for this connection, a system message could be sent every 0.0819s, or approximately 12 times every seconds. It will be apparent to one skilled in the art that other values for the maximum bandwidth percentage are possible. It is not the intention of the inventor to limit the maximum percentage to any example provided.

[00197] The recommended priority to be given to each message is now provided. System messages at the same priority level will be concatenated together. Messages for establishing and deleting HSPPs, including messages for marking HSPPs as active or not active, HSPP tokens for flow control for HSPPs and HSPP markers, are high priority messages. Medium high priority messages include loop test GUIDs, standard flow control messages such as token updates and additional quota requests. Latency

updates for queue propagation or when the node latency of a queue on a node 10 goes to infinity or changes from infinity to non-infinity are also medium high priority when these latency updates are on an HSPP.

[00198] The remaining system messages are low priority and thus must share a percentage of the maximum bandwidth available. However, among these low priority messages, there are also a number of differing classes. These are reflected in the percentage of the percentage of the maximum bandwidth for system messages allocated to each class of low priority messages. For example, if a class of low priority messages is allocated 80 per cent of the percentage of maximum bandwidth available for system messages and this latter percentage is 5 per cent, then messages of this class of low priority messages receives a total of four percent of the maximum available bandwidth. Messages that fall in the same class of low priority messages are concatenated together by the NCM 20.

[00199] The first class of low priority messages is allocated a recommended percentage of 80 % by the NCM 20 of the bandwidth allocated for low priority messages. These messages include latency updates for queue propagation and latency updates that are emitted when the node latency for a queue on a node goes to infinity or from infinity to a non-infinite value, provided the latency update is not being sent on an HSPP. Latency updates for queues that are currently transmitting EUM messages also fall in this class. Finally, whenever the NCM 20 on a node 10 updates it's target nodes 10 and needs to inform neighbor nodes 10, such as for testing instant loops, these updates are also in this class.

[00200] The next class of low priority system messages receives a recommended percentage of 15 % of the bandwidth allocated for low priority messages. These messages include round robin latency updates for all queues that were recently broadcasting EUM messages or for queues that are within the top 5 percent of all queues in terms of greatest change in node latency. Finally, the last class of low priority messages receives a

recommended five percent of the bandwidth for low priority messages. This class consists of round robin updates for all queues.

[00201] A further description of an embodiment of the invention is provided in Appendix "A".

[00202] It will be apparent to one skilled in the art that various modifications to the invention and embodiment described herein are possible without departing from the spirit and scope of the invention. Accordingly, it is to be understood that the present invention has been described by way of illustration and not limitation.

I CLAIM:

1. A system for transmission of messages between nodes on a network, said system comprising:

(a) a plurality of queues on each node; and

(b) a network communication manager on each node, wherein said network communication manage has knowledge of neighbour nodes and knowledge of all queues on each node.

2. A method for determining the best path through a network comprising the steps of:

(a) determining the latency and capacity of neighbour nodes and selecting the most efficient neighbour node to receive a message; and

(b) repeating step (a) on a regular basis.

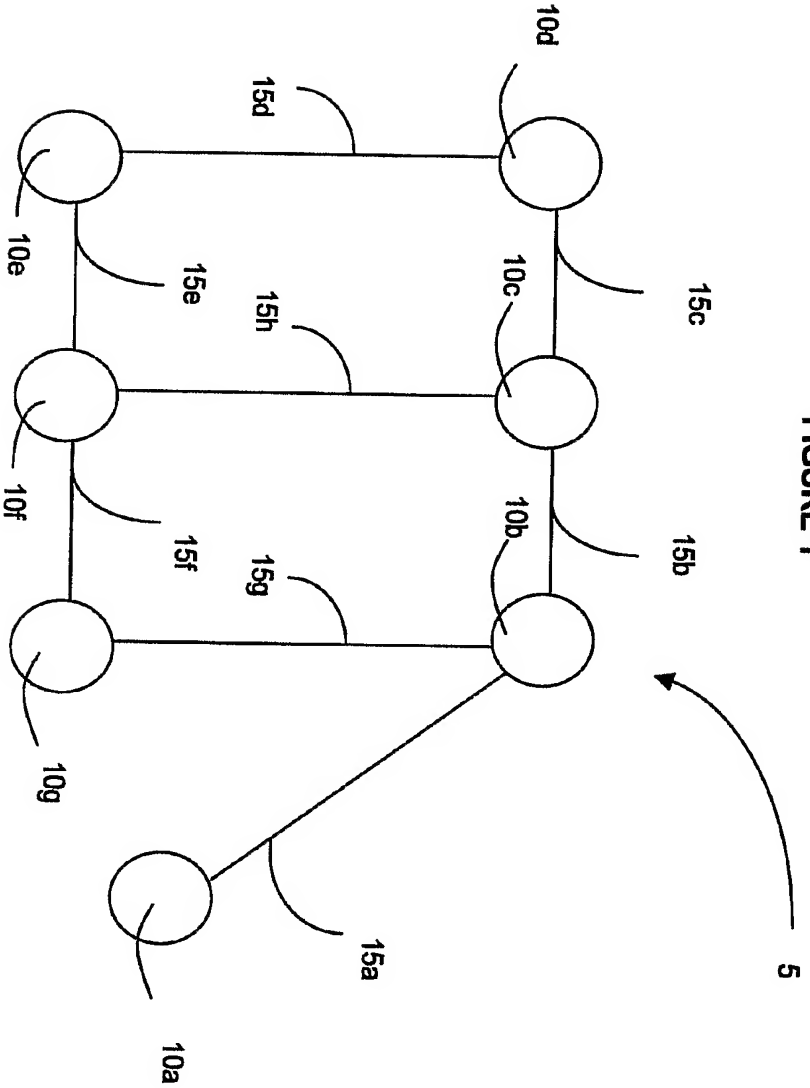

FIGURE 1

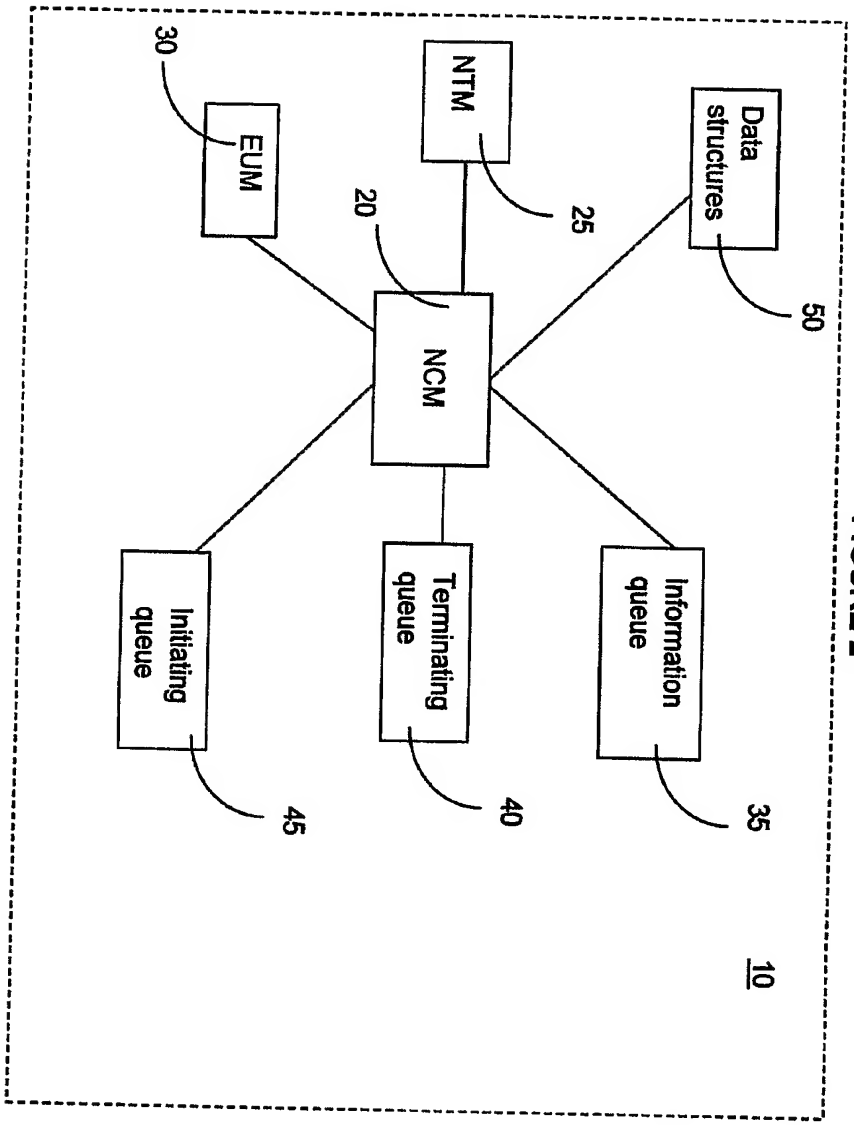
Data structures
50
NTM
25
EUM
30
20
NCM
Information queue
35
Terminating queue
40
Initiating queue
45
10

FIGURE 2

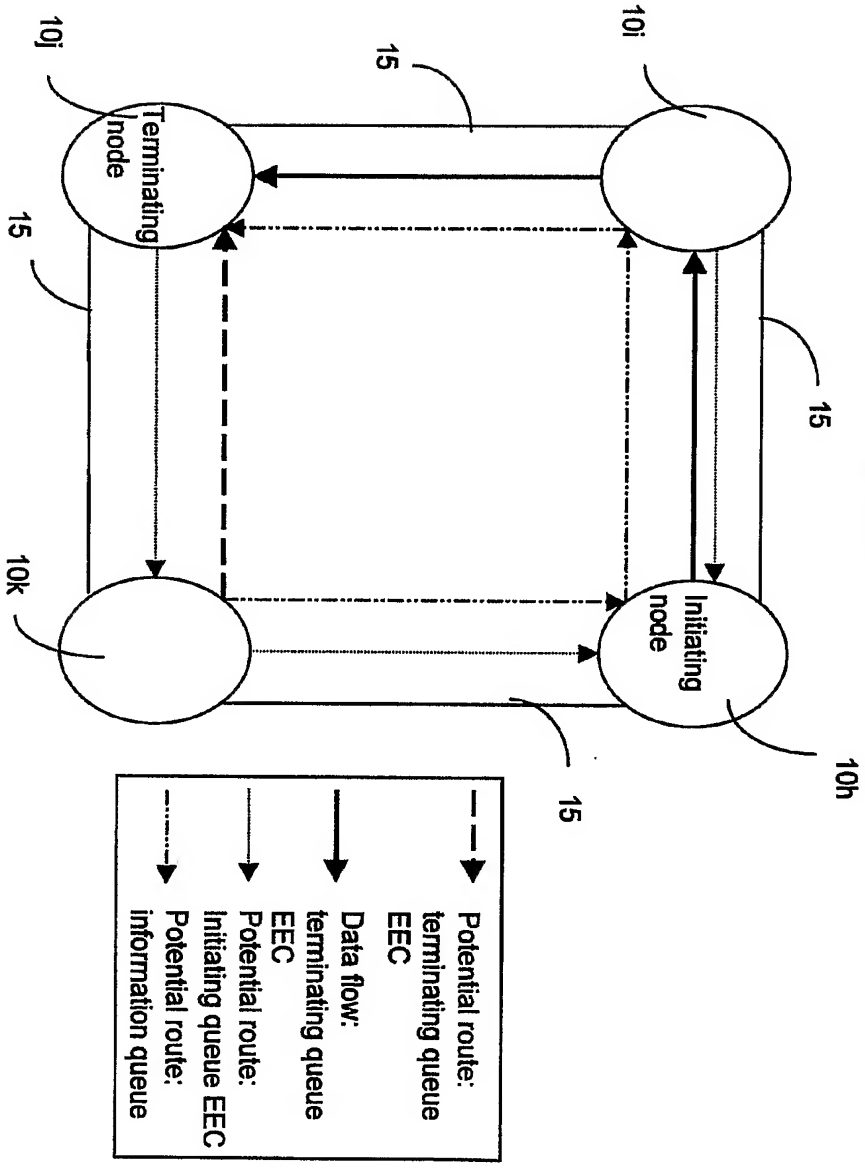
10i
15
10j
Terminating node
15
15
10k
Initiating node
10h
15
Potential route: terminating queue EEC
Data flow: terminating queue EEC
Potential route: Initiating queue EEC
Potential route: information queue

FIGURE 3

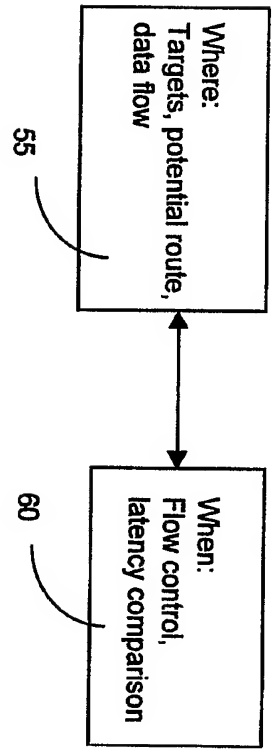
Where:
Targets, potential route,
data flow
55
When:
Flow control,
latency comparison
60

FIGURE 4

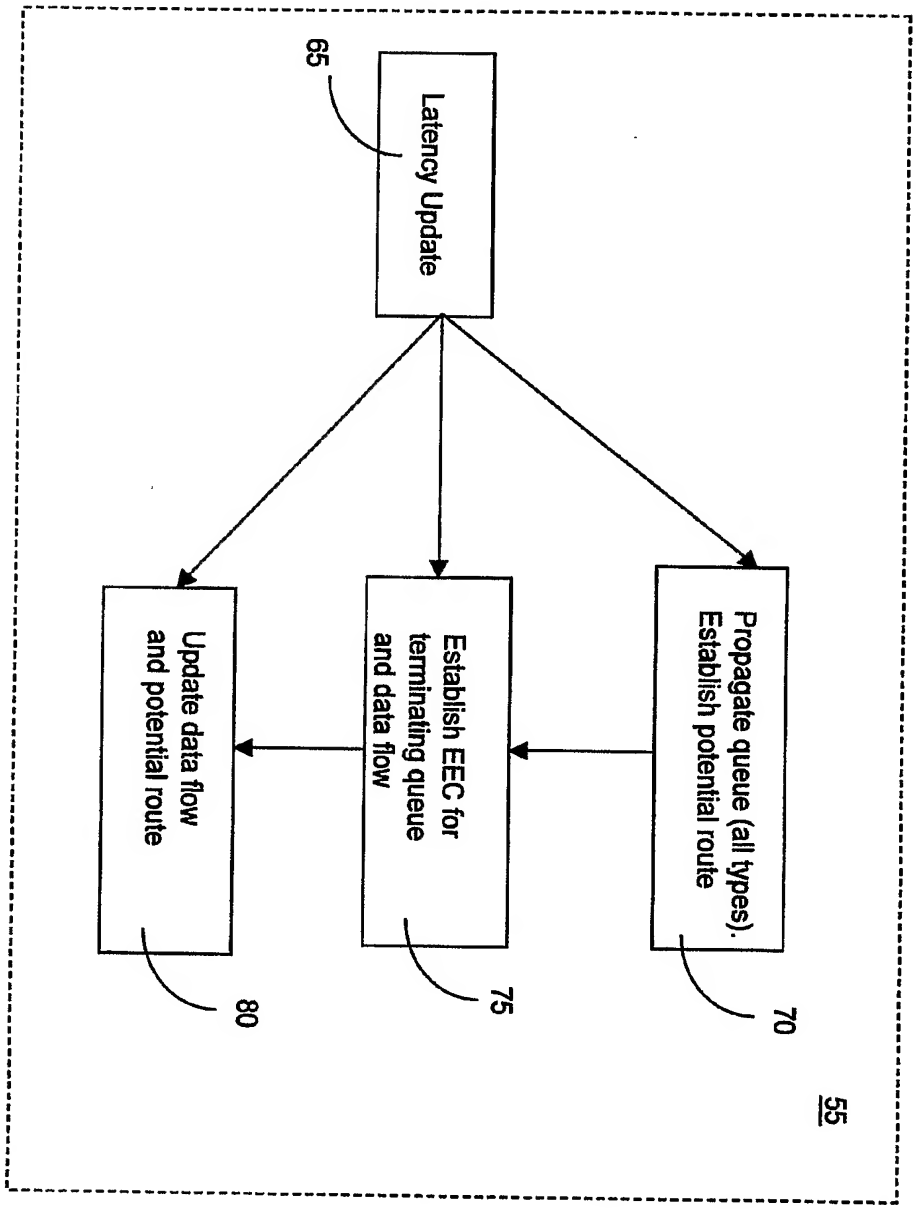
55
65
Latency Update
70
Propagate queue (all types). Establish potential route
75
Establish EEC for terminating queue and data flow
80
Update data flow and potential route

FIGURE 5

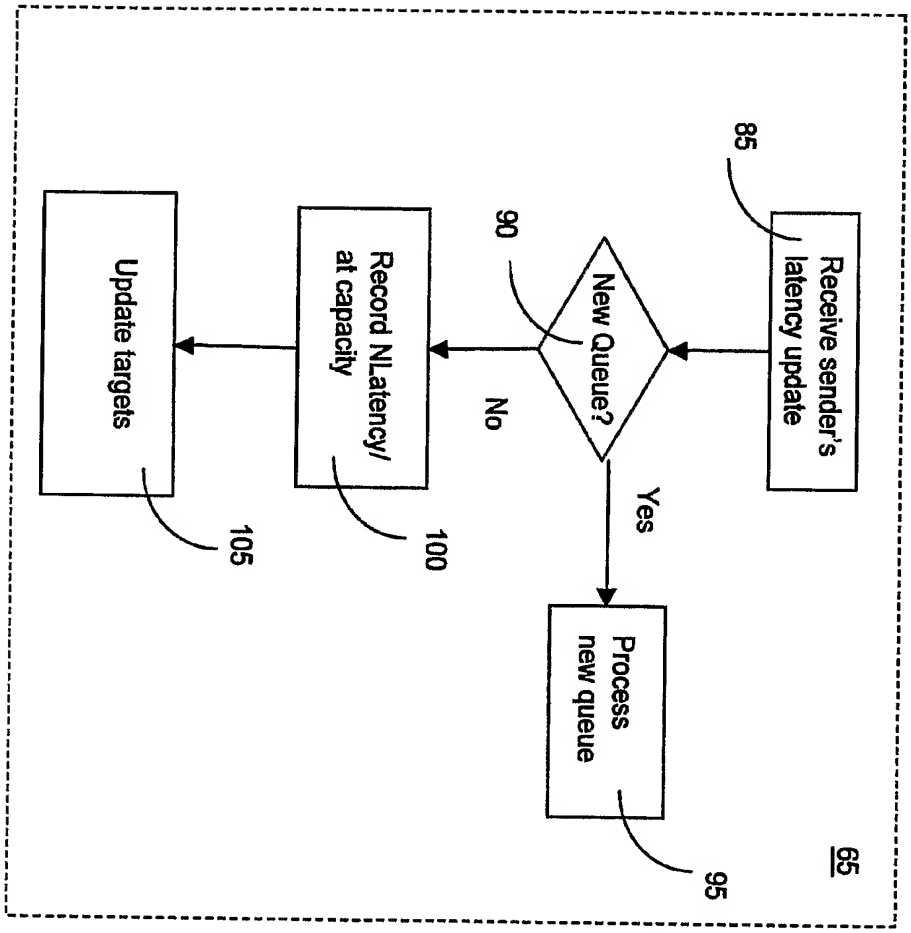
Receive sender's latency update
85
New Queue?
90
Yes
No
Process new queue
95
Record NLatency/ at capacity
100
Update targets
105
65

FIGURE 6

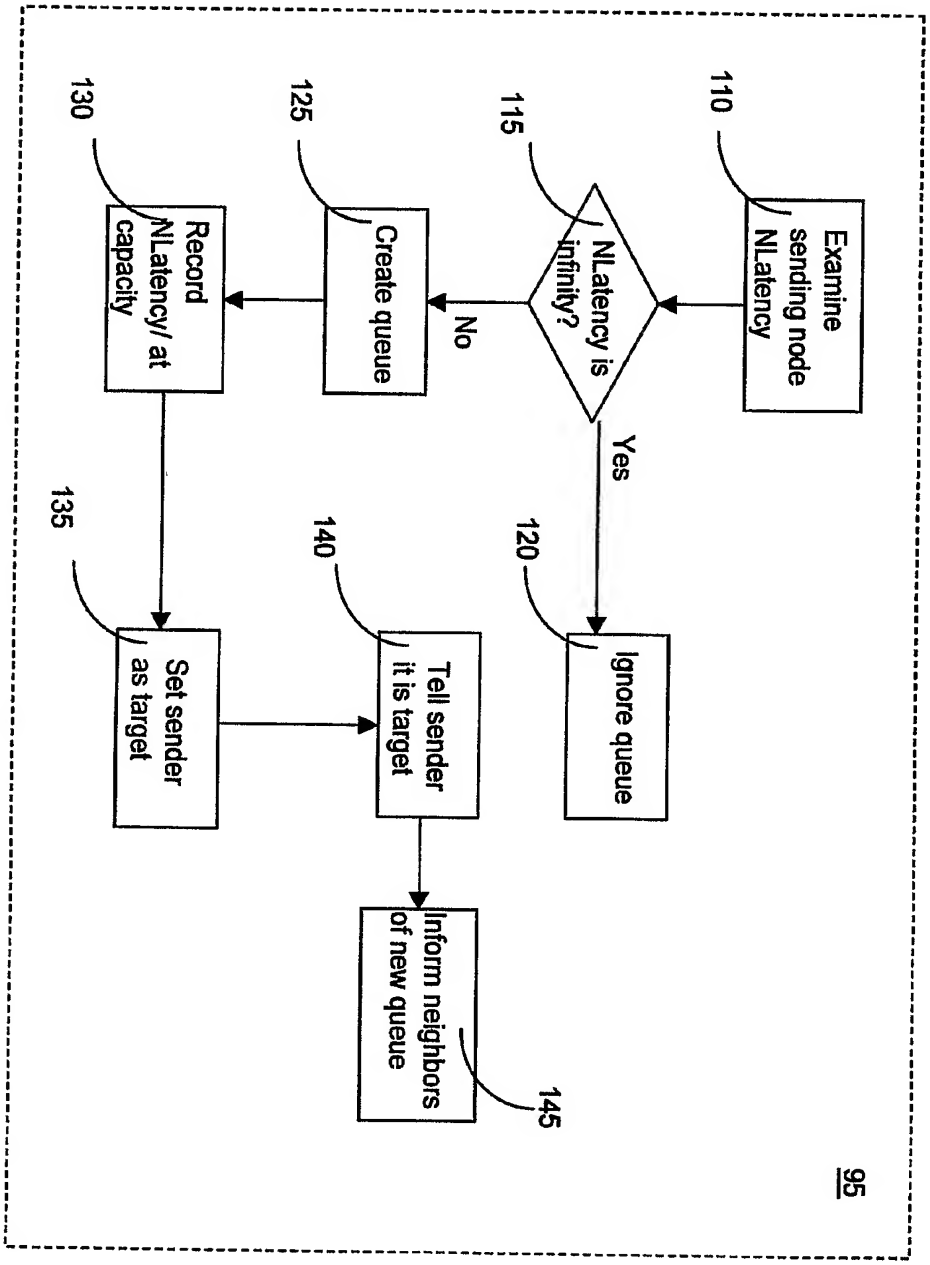
95
110
Examine sending node NLatency
115
NLatency is infinity?
Yes
No
120
Ignore queue
125
Create queue
130
Record NLatency/ at capacity
135
Set sender as target
140
Tell sender it is target
145
Inform neighbors of new queue

FIGURE 7

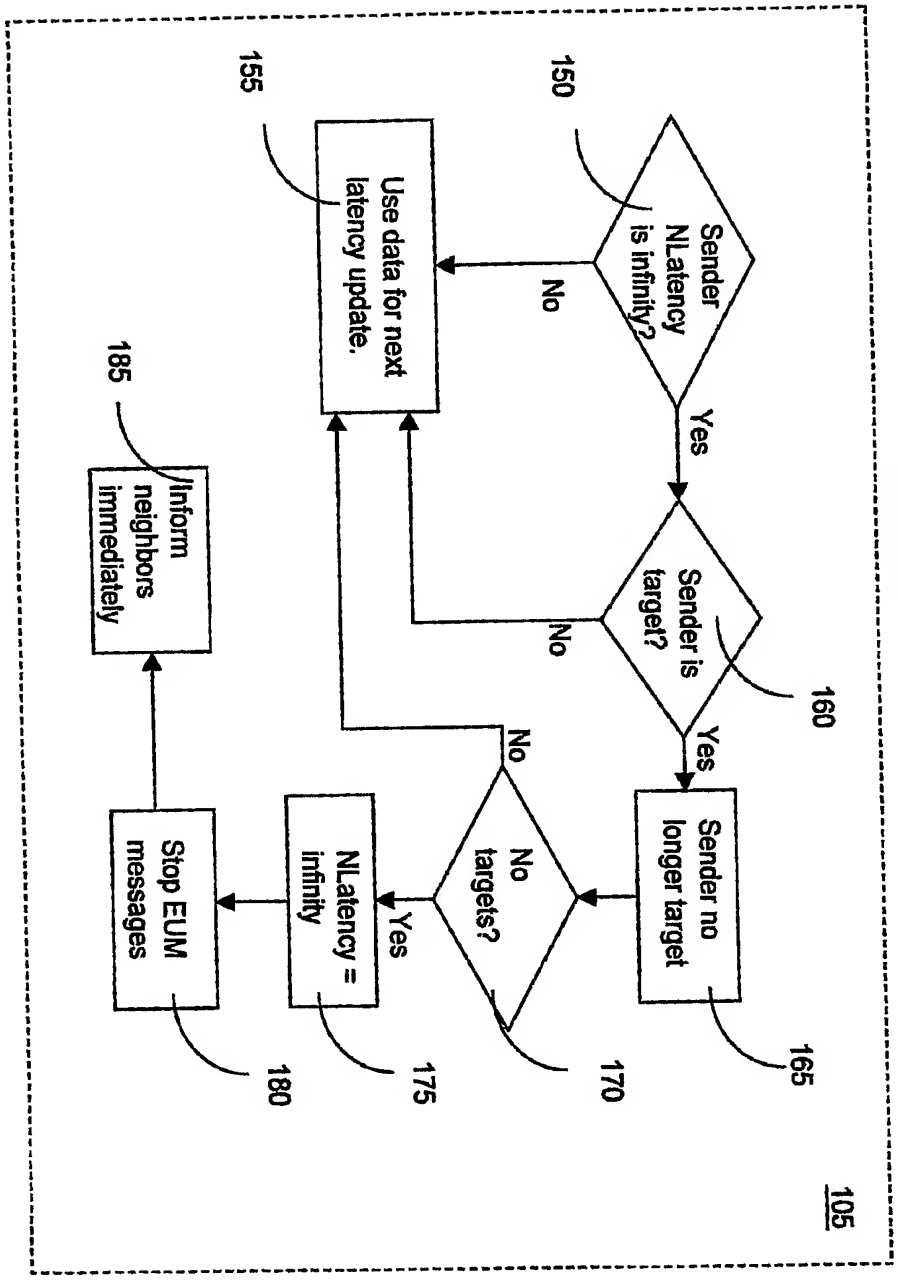
105
Sender NLatency is infinity?
150
No
Yes
Use data for next latency update.
155
Sender is target?
160
No
Yes
Sender no longer target
165
No targets?
170
No
Yes
NLatency = infinity
175
Stop EUM messages
180
Inform neighbors immediately
185

FIGURE 8

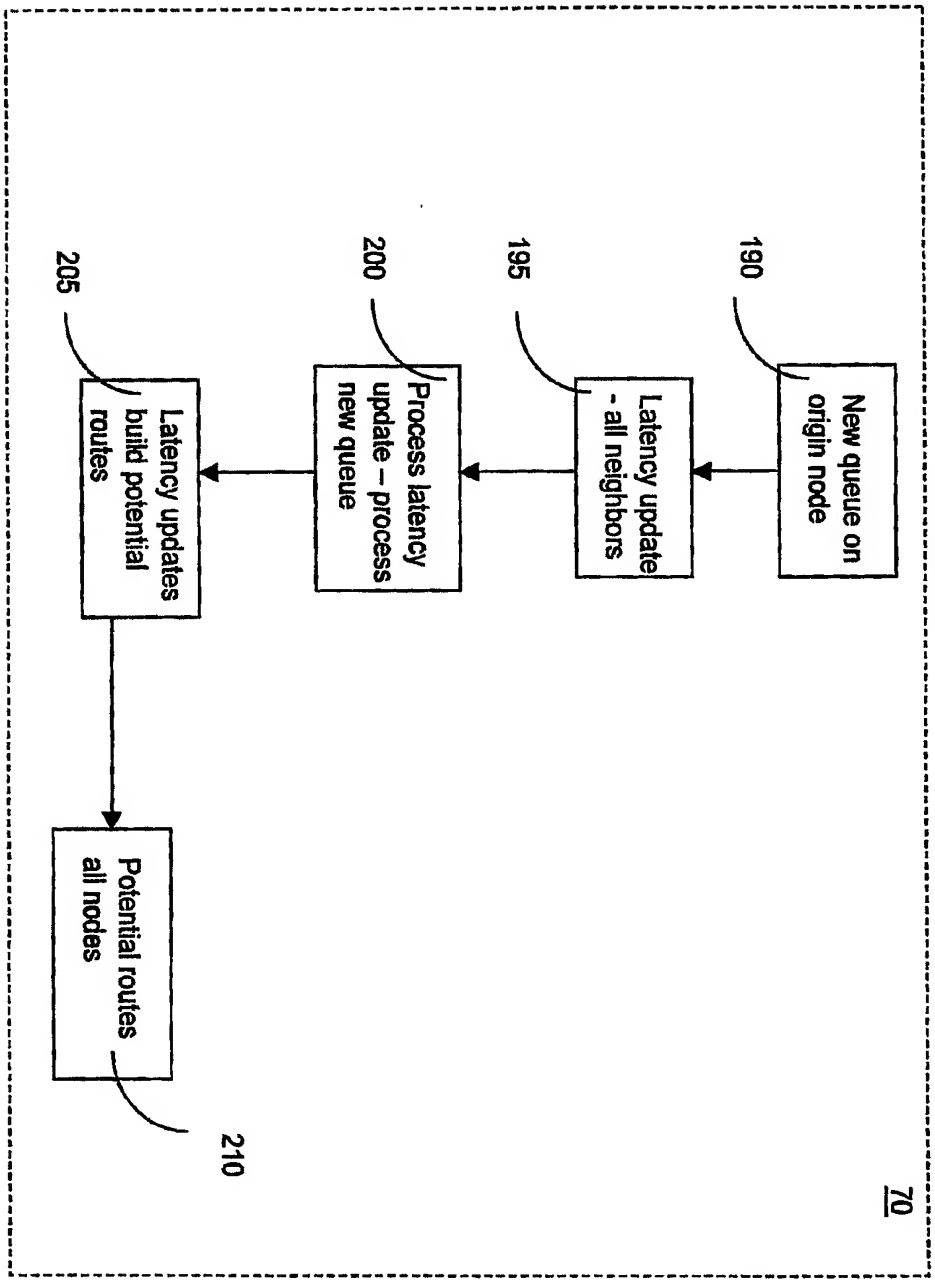
70
New queue on origin node
190
Latency update - all neighbors
195
Process latency update – process new queue
200
Latency updates build potential routes
205
Potential routes all nodes
210

FIGURE 9

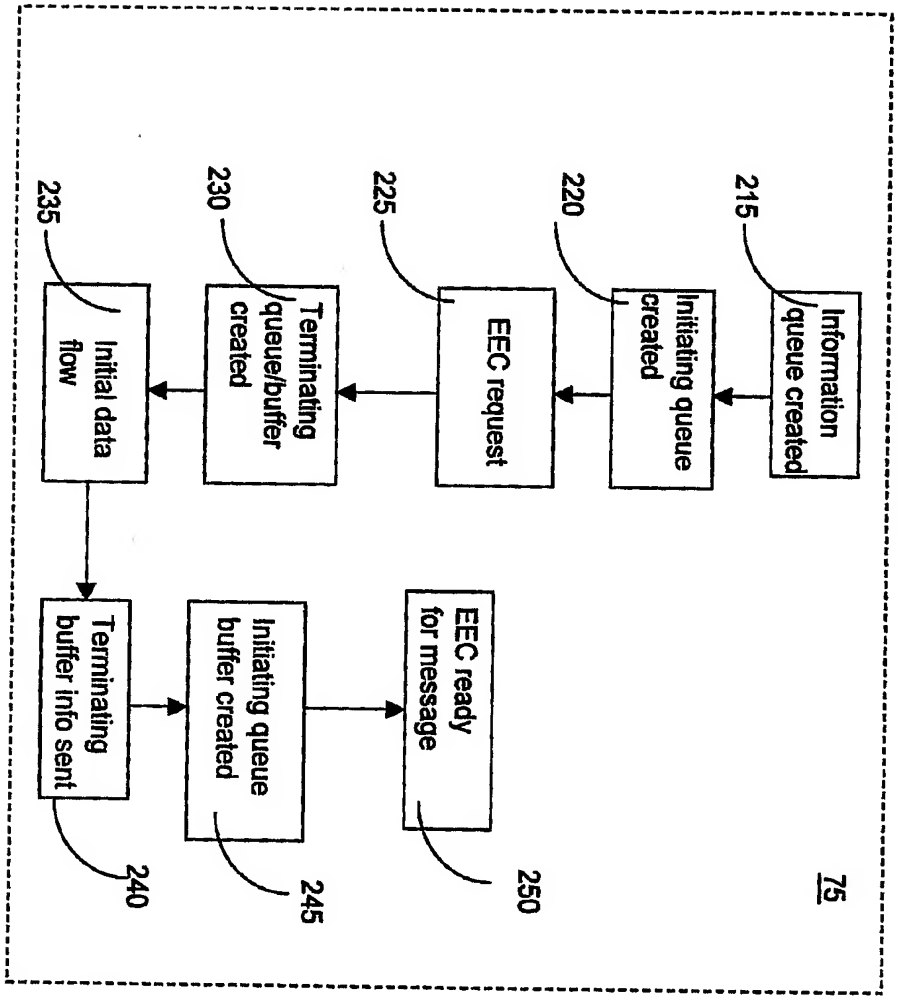
75
Information queue created
215
Initiating queue created
220
EEC request
225
Terminating queue/buffer created
230
Initial data flow
235
Terminating buffer info sent
240
Initiating queue buffer created
245
EEC ready for message
250

FIGURE 10

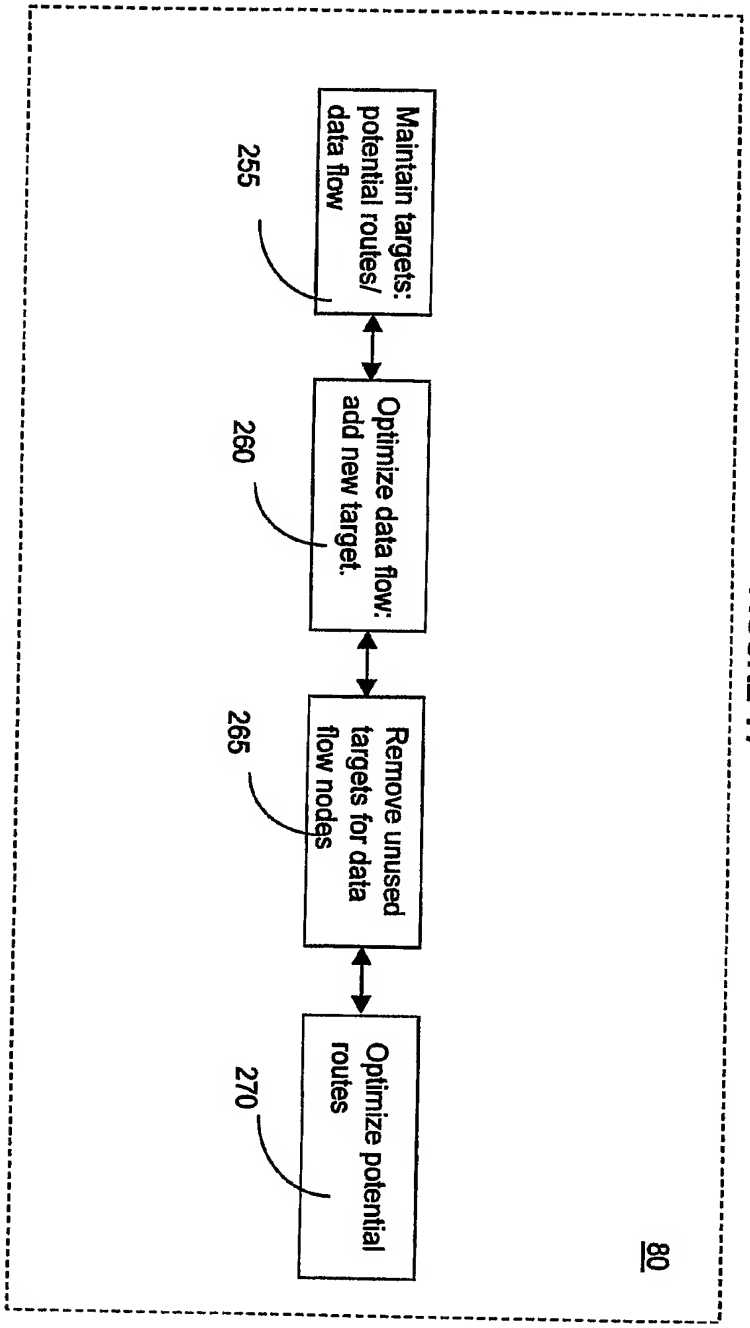
Maintain targets: potential routes/ data flow
255
Optimize data flow: add new target.
260
Remove unused targets for data flow nodes
265
Optimize potential routes
270
80

FIGURE 11

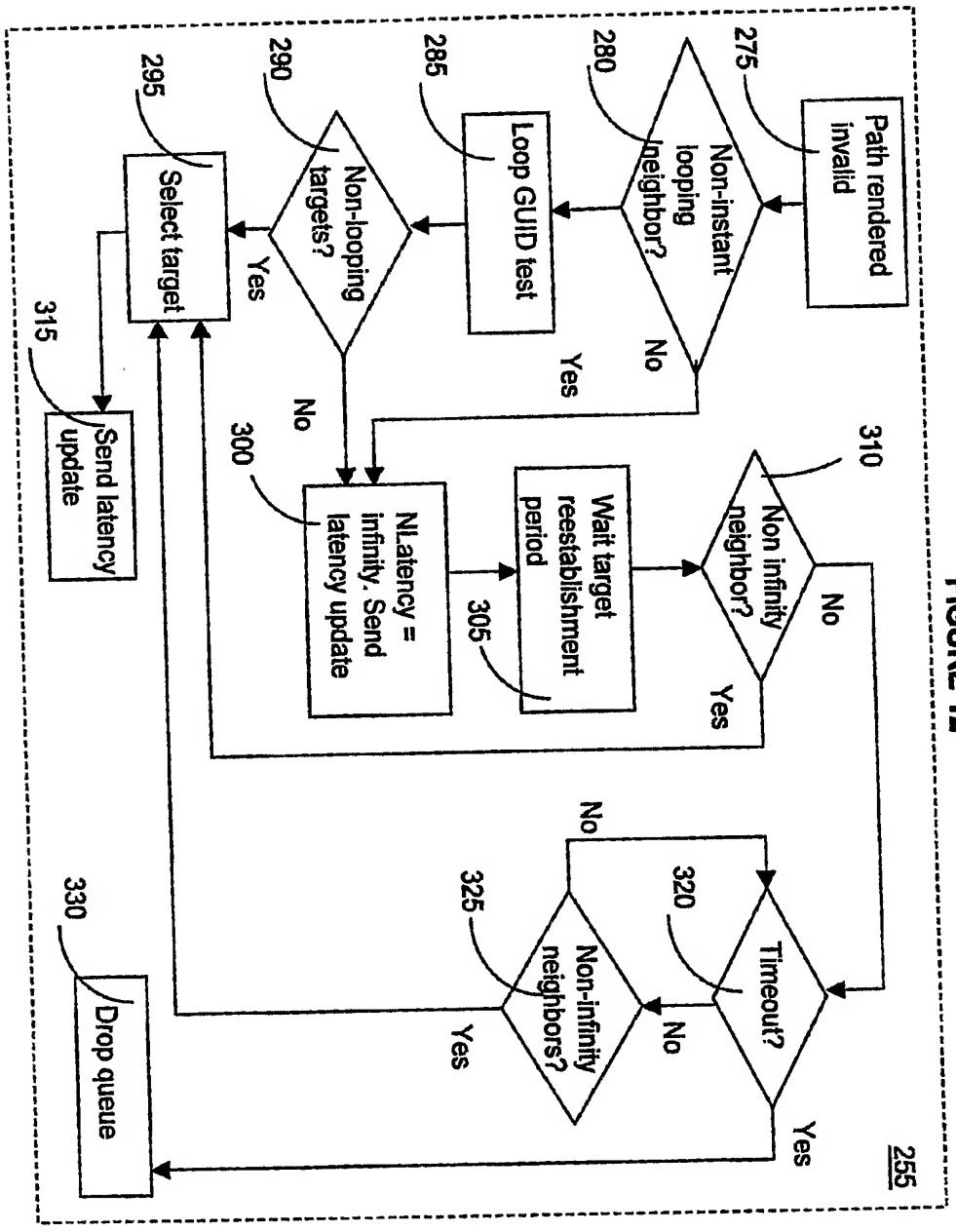
Path rendered invalid
275
Non-instant looping neighbor?
280
No
Yes
Loop GUID test
285
Non-looping targets?
290
Yes
No
Select target
295
Send latency update
315
NLatency = infinity. Send latency update
300
Wait target reestablishment period
305
Non infinity neighbor?
310
No
Yes
Timeout?
320
No
Yes
Non-infinity neighbors?
325
No
Yes
Drop queue
330
255

FIGURE 12

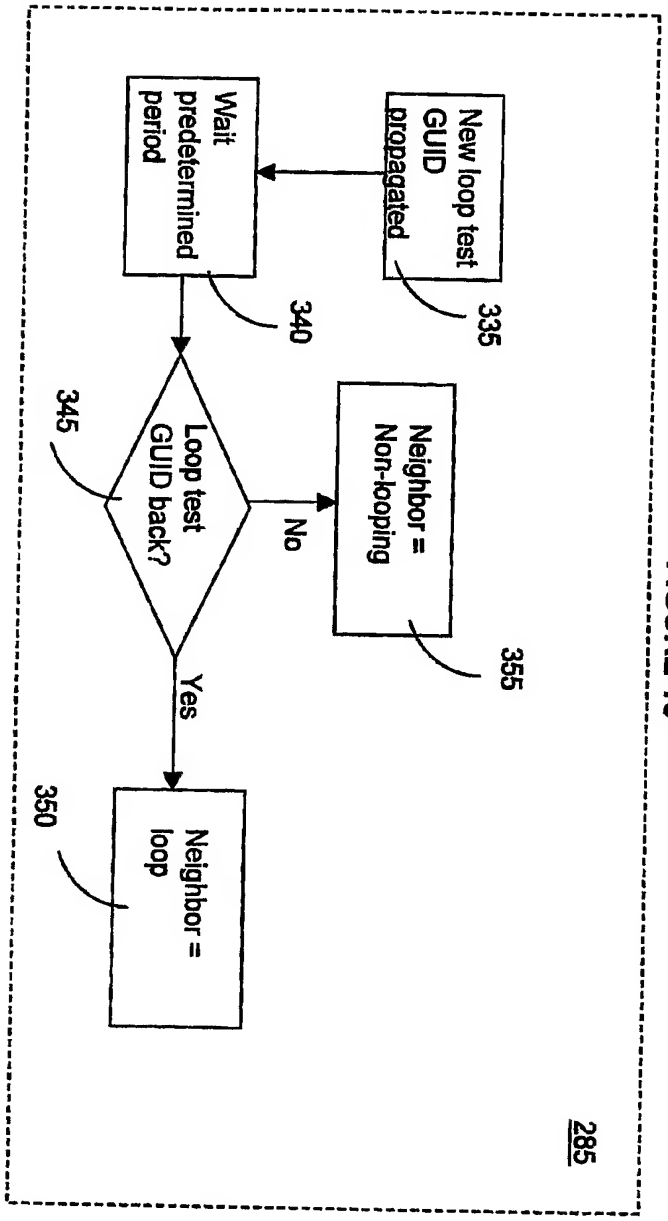
285
New loop test GUID propagated
335
Wait predetermined period
340
Loop test GUID back?
345
No
Neighbor = Non-looping
355
Yes
Neighbor = loop
350

FIGURE 13

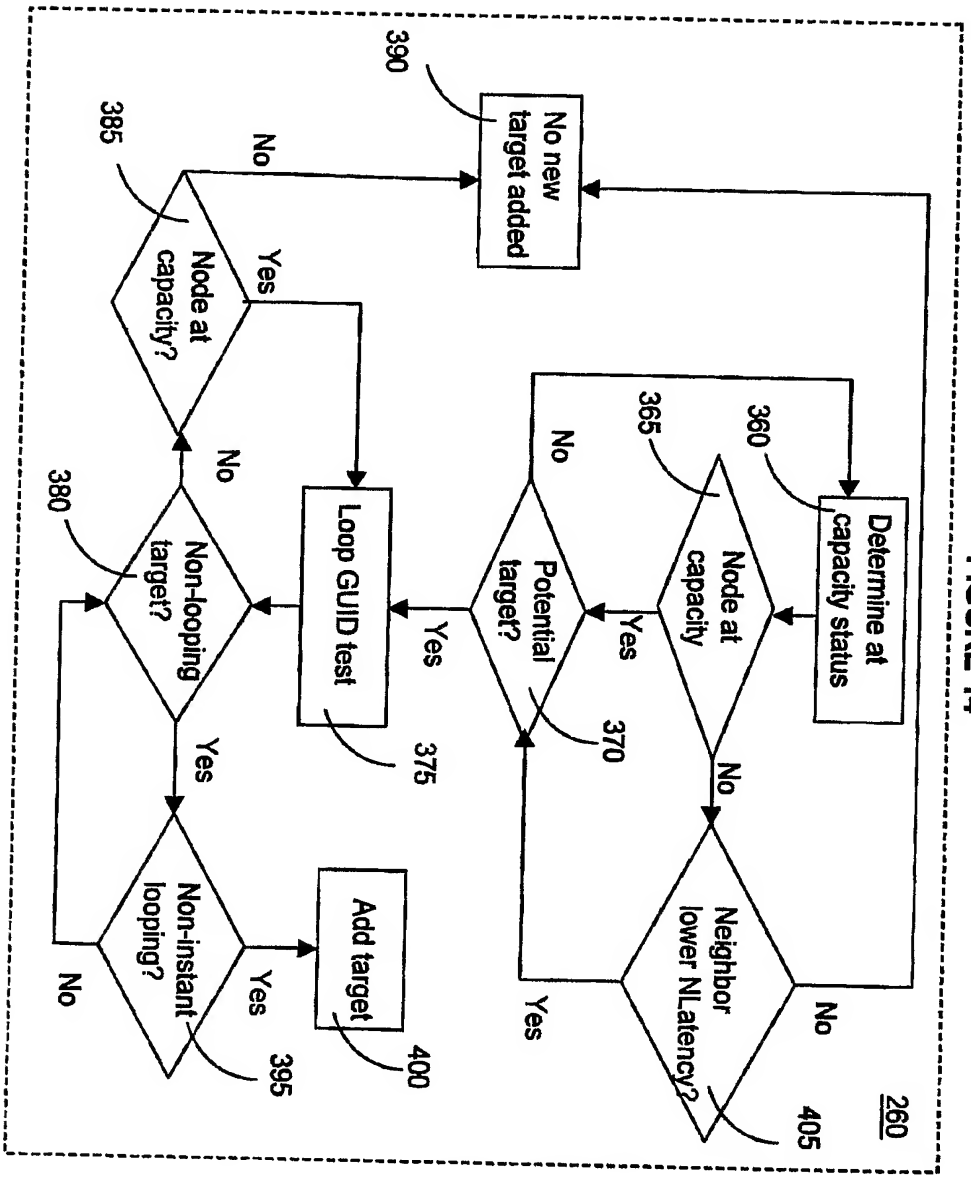
260
Determine at capacity status
360
Node at capacity
365
Yes
No
Potential target?
370
No
Yes
Neighbor lower NLatency?
405
No
Yes
No new target added
390
Node at capacity?
385
No
Yes
Loop GUID test
375
Non-looping target?
380
No
Yes
Non-instant looping?
395
No
Yes
Add target
400

FIGURE 14

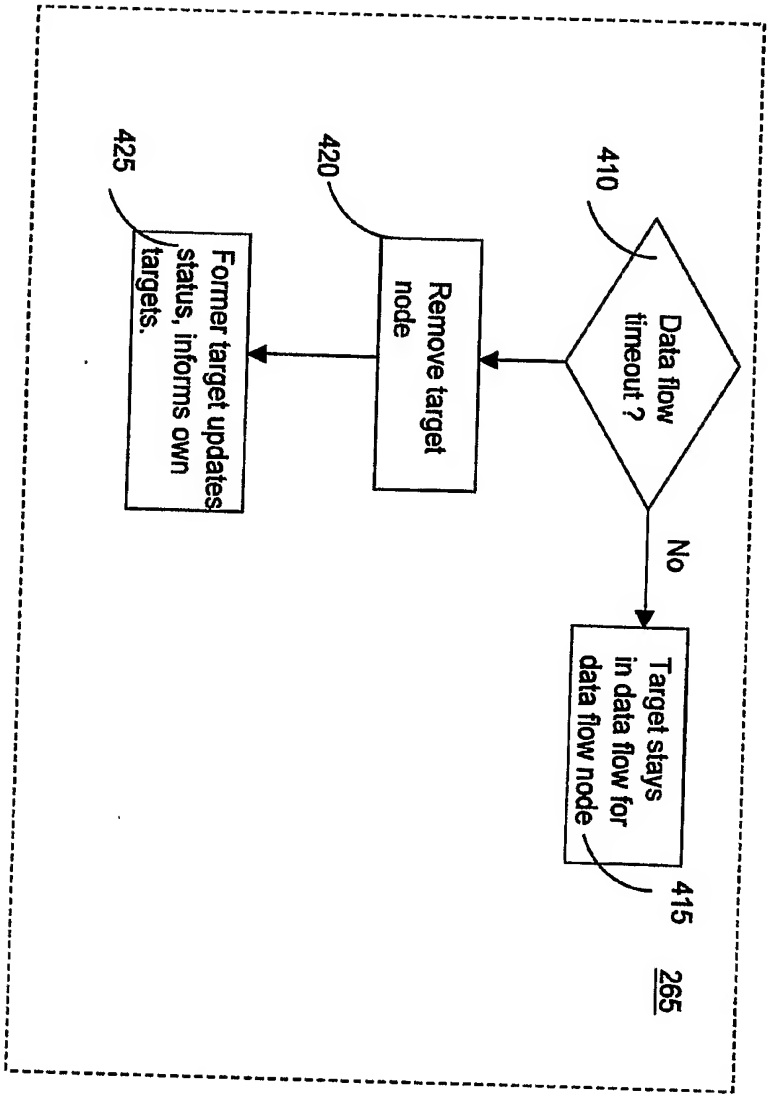
Data flow timeout ?
410
No
Target stays in data flow for data flow node
415
265
Remove target node
420
Former target updates status, informs own targets.
425

FIGURE 15

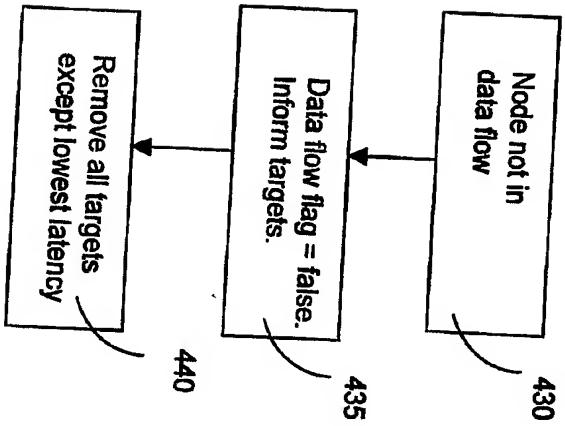
Node not in data flow
430
Data flow flag = false. Inform targets.
435
Remove all targets except lowest latency
440

FIGURE 16

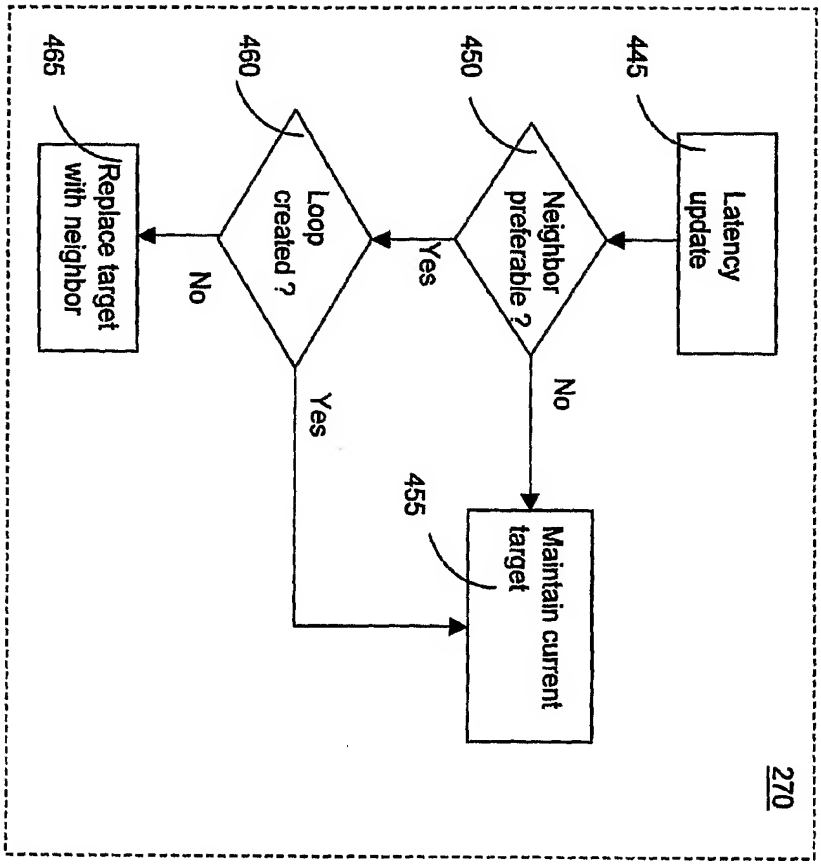
Latency update
445
Neighbor preferable ?
450
Yes
No
Loop created ?
460
Yes
No
Replace target with neighbor
465
Maintain current target
455
270

FIGURE 17

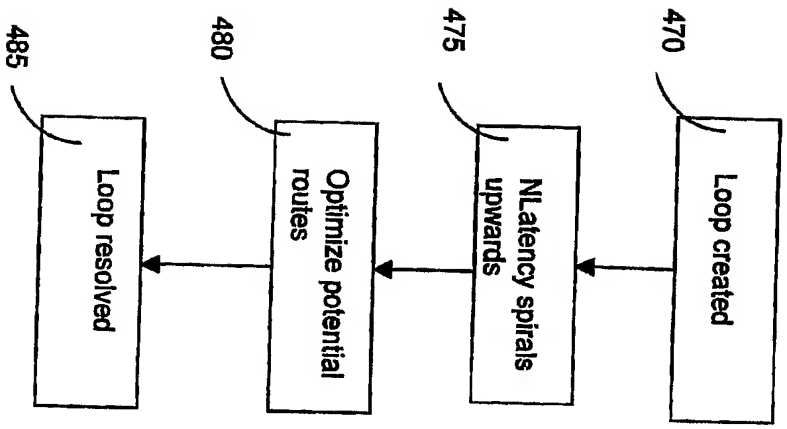
470
Loop created
475
NLatency spirals upwards
480
Optimize potential routes
485
Loop resolved

FIGURE 18

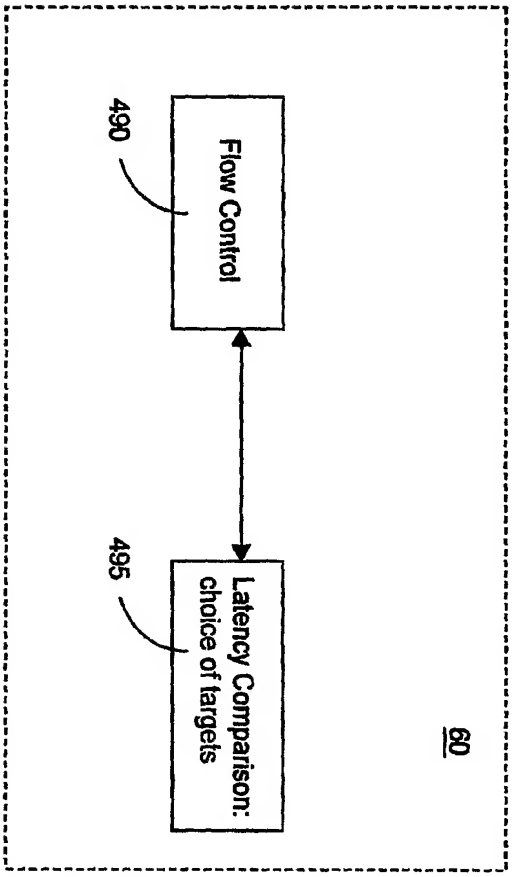
Flow Control
490
Latency Comparison: choice of targets
495
60

FIGURE 19

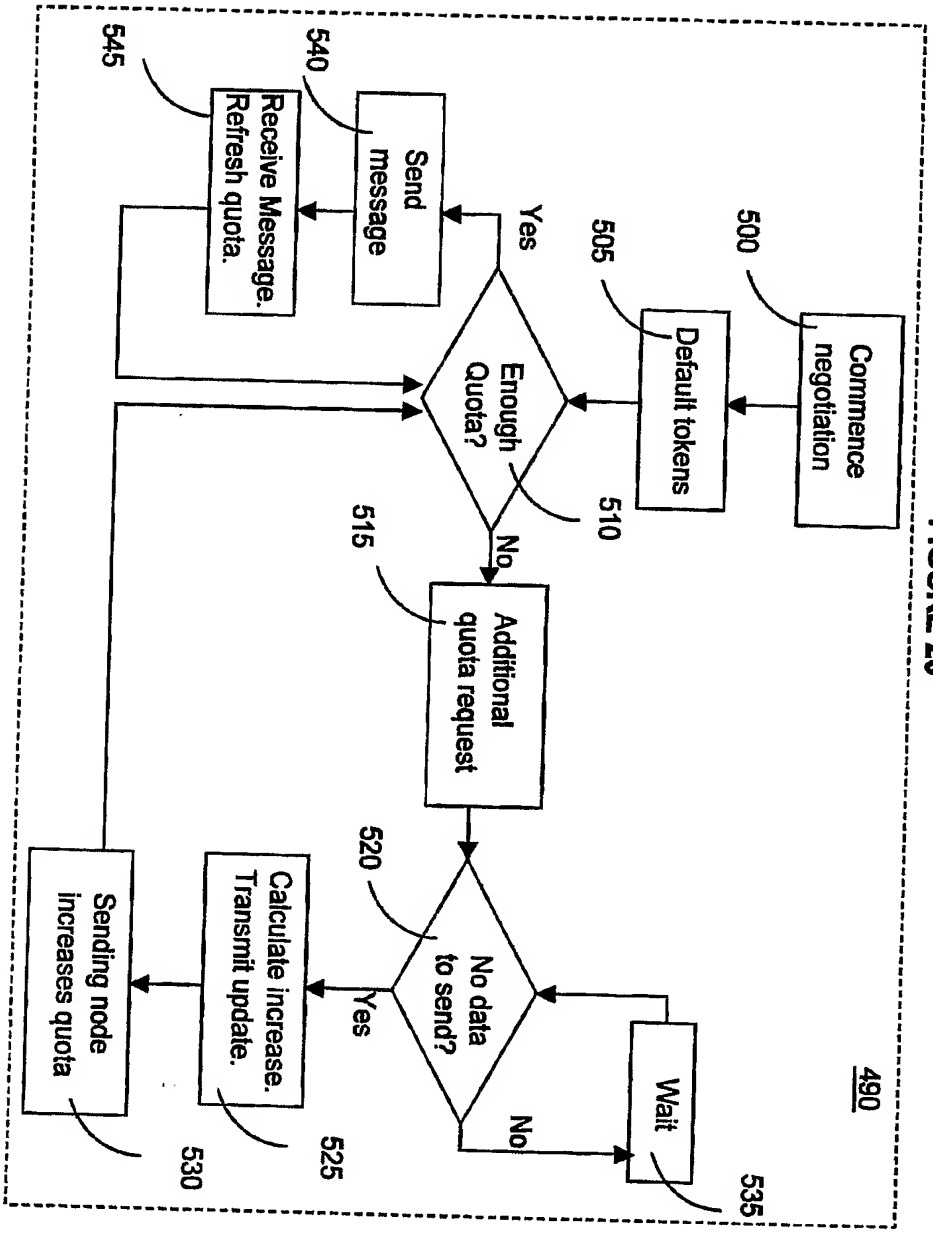
490
Commence negotiation
500
Default tokens
505
Enough Quota?
510
Yes
No
Send message
540
Receive Message. Refresh quota.
545
Additional quota request
515
No data to send?
520
Yes
No
Calculate increase. Transmit update.
525
Sending node increases quota
530
Wait
535

FIGURE 20

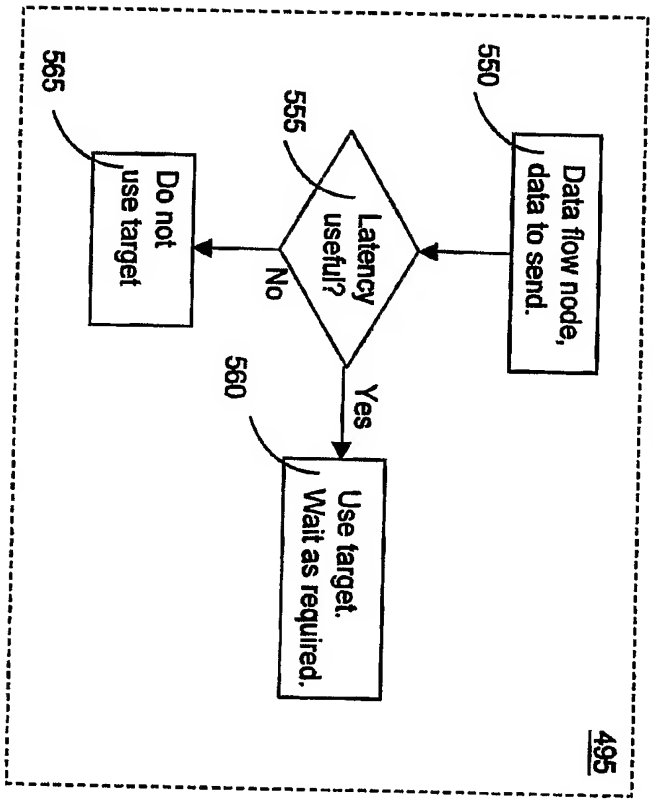
Data flow node, data to send.
550
Latency useful?
555
No
Yes
Do not use target
565
Use target. Wait as required.
560
495

FIGURE 21